AD-A099 125　DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC　F/G 5/2
PAGED GIRS (GRAPH INFORMATION RETRIEVAL SYSTEM) USERS MANUAL.(U)
MAY 81　I S ZARITSKY
UNCLASSIFIED　DTNSRDC-81/024　NL

1 OF 2
AD A
099125

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

REPORT DOCUMENTATION PAGE		READ INSTRUCTIONS BEFORE COMPLETING FORM
1. REPORT NUMBER DTNSRDC-81/024	2. GOVT ACCESSION NO. AD-A099 125	3. RECIPIENT'S CATALOG NUMBER
4. TITLE (and Subtitle) PAGED GIRS (GRAPH INFORMATION RETRIEVAL SYSTEM) USERS MANUAL		5. TYPE OF REPORT & PERIOD COVERED Final 6. PERFORMING ORG. REPORT NUMBER
7. AUTHOR(s) Irving S. Zaritsky		8. CONTRACT OR GRANT NUMBER(s) F43411
9. PERFORMING ORGANIZATION NAME AND ADDRESS David W. Taylor Naval Ship Research and Development Center Bethesda, Maryland 20084		10. PROGRAM ELEMENT, PROJECT, TASK AREA & WORK UNIT NUMBERS (See reverse side)
11. CONTROLLING OFFICE NAME AND ADDRESS Naval Sea Systems Command Washington, D.C. 20362		12. REPORT DATE May 1981 13. NUMBER OF PAGES 156
14. MONITORING AGENCY NAME & ADDRESS(if different from Controlling Office)		15. SECURITY CLASS. (of this report) UNCLASSIFIED 15a. DECLASSIFICATION/DOWNGRADING SCHEDULE

16. DISTRIBUTION STATEMENT (of this Report)

APPROVED FOR PUBLIC RELEASE: DISTRIBUTION UNLIMITED

17. DISTRIBUTION STATEMENT (of the abstract entered in Block 20, if different from Report)

18. SUPPLEMENTARY NOTES

19. KEY WORDS (Continue on reverse side if necessary and identify by block number)

Associative Memory	Hashed Addressing
Data Base	Information Retrieval System
Data Definition Language	Data Base Management System
Data Management	Paging Schemes
Graph	

20. ABSTRACT (Continue on reverse side if necessary and identify by block number)

The report describes the use of the paged version of an associative (content addressable) computer memory simulation called GIRS (Graph Information Retrieval System). GIRS provides a convenient and efficient technique for the dynamic insertion, retrieval, modification, and deletion of data in a data base. Pointer manipulation is convenient and paged GIRS is well adapted for concurrent operation on more than one graph and therefore will

(Continued on reverse side)

UNCLASSIFIED

SECURITY CLASSIFICATION OF THIS PAGE (When Data Entered)

381012

(Block 10)

Program Element 62543N
Project F43411
Task Area ZF 43411001
Work Unit 1808-009

(Block 20 continued)

handle shared and distributed data bases. Users of a large data base could have their own unique description of common data which might be stored elsewhere. The paged version of GIRS allows for a wide range of flexibilities, in which, at a minimum, a user may leave many parameters to default. For maximum flexibility, however, a user may include a user-embedded strategy and hence may satisfy queries of various degrees of imprecision depending on the inferential search technique used.

The implementation described here is in FORTRAN for the PDP 11/45 computer system and is described in sufficient detail to allow conversion to another computer or alteration of the existing overlay structure.

TABLE OF CONTENTS

LIST OF TABLES

ABSTRACT

This report describes the use of the paged version of an associative (content addressable) computer memory simulation called GIRS (Graph Information Retrieval System). GIRS provides a convenient and efficient technique for the dynamic insertion, retrieval, modification, and deletion of data in a data base. Pointer manipulation is convenient and paged GIRS is well adapted for concurrent operation on more than one graph and therefore will handle shared and distributed data bases. Users of a large data base could have their own unique description of common data which might be stored elsewhere. The paged version of GIRS allows for a wide range of flexibilities, in which, at a minimum, a user may leave many parameters to default. For maximum flexibility, however, a user may include a user-embedded strategy and hence may satisfy queries of various degrees of imprecision depending on the inferential search technique used.

The implementation described here is in FORTRAN for the PDP 11/45 computer system and is described in sufficient detail to allow conversion to another computer or alteration of the existing overlay structure.

ADMINISTRATIVE INFORMATION

This work was completed in the Computer Science Division of the Computation, Mathematics, and Logistics Department under the sponsorship of NAVSEA 03F, Task Area ZF 43411001, Work Unit 1808-009.

INTRODUCTION

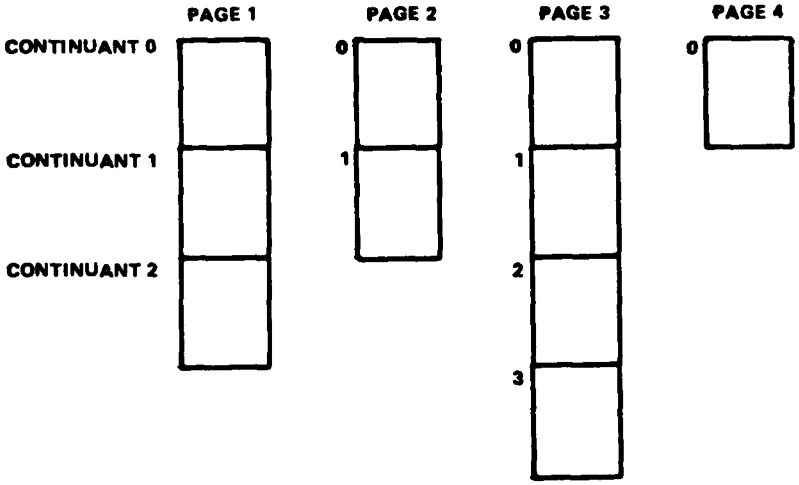
This report describes the use, and to a lesser degree, the implementation of the paged version of an associative (content addressable) computer memory simulation called GIRS (Graph Information Retrieval System). GIRS provides a convenient and efficient technique for the dynamic insertion, retrieval, modification, and deletion of data in a data base. Pointer manipulation is convenient and paged GIRS is well adapted to handle shared and distributed data bases. The flexibility of the paged version of GIRS allows the user the option of leaving many parameters to default. For maximum flexibility, however, a user may include a user-embedded strategy and hence may satisfy queries of various degrees of imprecision depending on the inferential search technique used.

The implementation described here is in FORTRAN for the PDP 11/45 computer system and is in sufficient detail to allow conversion to another computer or alteration of the existing overlay structure.

Paged or Out-Core GIRS is an extension of an existing version of GIRS as described by Zaritsky.[1*] The original version will be referred to in this report as "In-Core" GIRS. Out-Core GIRS is an implementation based on a report by Berkowitz,[2] "Design Trade-Offs For A Software Associative Memory." Some familiarity with In-Core GIRS is assumed and it is also assumed that the reader has access to copies of both prior reports.

BRIEF DESCRIPTION

With Out-Core GIRS, a graph may be segmented and placed onto as many as 63 logically and physically separate regions called pages. Pages can be extended in length--i.e., in the number of associations stored, but not in the number of addresses--as needed by a specified increment, called a continuant. Each page, as requested, contains one or more continuants (logical records of uniform physical length) as illustrated in the following diagram:



[*] A complete listing of references is given on page 149.

Continuants may be used to further partition a graph or merely to hold an overflow of data from a previous continuant. It is the continuant which is swapped from disk to the GIRS buffer and back. The user determines the continuant size and also the number of continuants which will reside in the GIRS buffer. The continuant size determines the maximum number of nodes and links which may be defined for each page.

If a user chooses a buffer size which holds only one continuant and requests just one page, then the system is similar to in-core GIRS except for the capability of automatic overflow to a new continuant.

MOTIVATION

The paged version of GIRS has several advantages over in-core GIRS and other data manipulation facilities:

1. Its large data storage capability (see the section entitled "Limitations and Memory Requirements").

2. Concurrent operation on more than one graph. Paged GIRS is ideal for shared and distributed data bases. Each user of a large data base might be assigned his or her own page to uniquely describe common data which might be stored elsewhere. An example of this type of application is described by Zaritsky.[3]

3. Its capability for a user-embedded strategy, which allows for the inclusion of operations such as an inferential search to handle imprecise queries. This capability is described in the section on "Adding a User-Embedded Strategy."

In the near future, a paged hardware associative memory[4] will be merged with out-core GIRS. The result will be an enhanced system with high speed relational processing.

MEMORY SCHEME

THE GIRS BUFFER

The GIRS buffer consists of four fields, represented by the four arrays, NODSPC, LSTSPC, LNKSPC, and FLGSPC, from commons LVVTR1, LVVTR2, LVVTR3, and LVVTR4, respectively, as was the case with the in-core version of GIRS. The buffer contains both continuants and a directory for locating the continuants residing in the buffer. The number of continuants that may reside in the buffer is unrestricted. It is fixed by the user in variable LVNCOR in labeled common LVBUFR. The buffer location immediately preceding the beginning of each continuant is called the control point (CP) and the directory is located at CP0.

THE CONTINUANT

Although each continuant requires an equal length of NODSPC, LSTSPC, LNKSPC, and FLGSPC,* each field is composed of three "spaces": a "working space" and an "available space," as was the case with in-core GIRS, and a space for the header to describe the state of the continuant. The header takes up eight cells,** two in each of the four fields. The header is described in Table 1; the variables (from labeled common LVHDVL) in brackets indicate the relative location from the beginning of the continuant.

*It is convenient to refer to the length of any one of these arrays, instead of all four, as being the continuant size. The continuant size requested by the user must be a multiple of 64 (up to 960 on default), but the actual usable continuant size is always two less than n*64 to account for the header.

**On the PDP 11, a cell takes up the space of one word.

TABLE 1 - THE CONTINUANT HEADER

NODSPC

a) The relative Mass Storage Address (MSA) of this continuant [THSMSA]

b) The REGister of Available SPace (REGASP) for this continuant [REGAS]

LSTSPC

a) This continuant's page number [PAGENO]

b) This continuant's relative position within the page [CONTNO]

LNKSPC

a) The size of this continuant's "working space" (number of insertions less deletions) [INSDEL]

b) The number of times this continuant has been accessed since it was last brought into main memory [USECT]

FLGSPC

a) [HDRFLG] The continuant descriptor flags, if on, indicate the following:

   1) LVWRIT - The continuant has been modified since it was brought into the buffer and therefore must be written out to disk when either another continuant is brought into the same segment (control point) of the buffer or when the file is closed.

   2) LVNUSE - The continuant has not yet been used. Either it has just been created or it has been brought into the buffer and not yet accessed. This flag is turned off when the continuant is accessed.

b) An indicator of how recently this continuant was brought into the buffer [READVL]

If a user wishes to access information described in Table 1 and if the desired continuant is the current continuant of the current page, then the user may do so by adding the bracketed label to LVCTRL and using that quantity as an index to the appropriate buffer array. For example, "Transfer to statement 10 if the continuant read into the buffer most recently (making it the current continuant of the current page) has not yet been accessed:"

```
IF((FLGSPC(LVCTRL + HDRFLG) .AND. LVNUSE) .NE. 0) GO TO 10.
```

THE DIRECTORY

The directory is a continuant without a header. It is never taken out of the buffer and is located at the beginning of the buffer. Its size (within each of the four fields of the buffer) is determined by the number of continuants residing in the buffer (LVNCOR) and is calculated as follows:

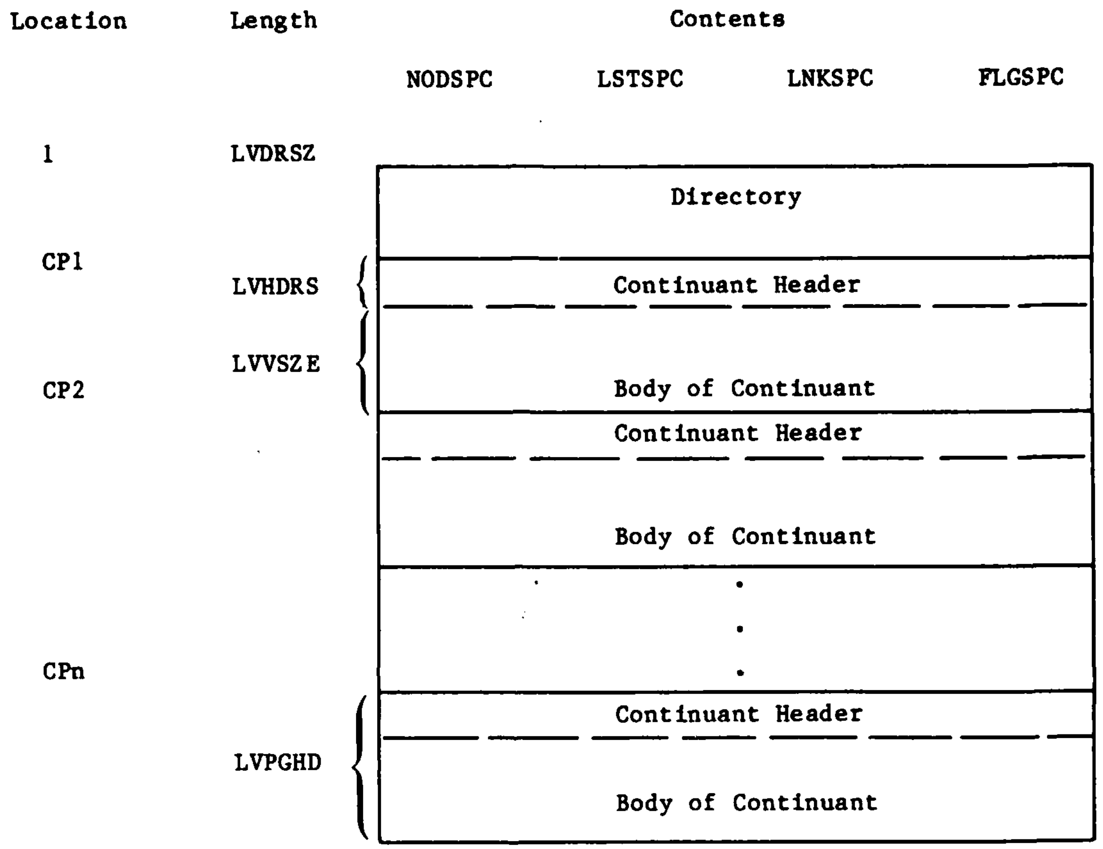
$$LVDRSZ = 64 * ((LVNCOR/64) + 1)$$

Each control point is stored as the sink node (integer value) of a triple where the source node is the "continuant number + 1" and the link is the page number. The directory address for the triple is determined as follows:

$$LOC = (\text{Page No.} + \text{Cont. No} + 1) \text{ Mod } (LVDRSZ)$$

THE BUFFER COMPOSITION

A typical buffer may be illustrated as follows:



where: CP1 = LVDRSZ

CP2 = LVDRSZ + LVHDRS + LVVSZE

CPn = LVDRSZ + (n-1) * LVPGHD

LVHDRS = 2

LVPGHD = LVHDRS + LVVSZE

LVVSZE = continuant size as defined by the user

INITIALIZATION OF THE GIRS BUFFER

As for In-Core GIRS, the buffer is initialized by calling subroutine LVSETP. The available space (AS) ring is identical to that in the In-Core version except that as many copies of it will be placed in the buffer as there are continuants residing in the buffer.

The values in the continuant header as described in Table 1 are initialized as follows:

NODSPC

a) The relative block address of the continuant on the disk file as computed by LVSETP

b) REGASP = 1

LSTSPC

a) The page number of the continuant

b) Continuant numbers as assigned in order of their creation, beginning with zero

LNKSPC

a) The number of spaces in the continuant which have been removed from AS = 0

b) Access count = 0

FLGSPC

a) Continuant descriptor flags = 0

b) Continuant I/O histoiy set to the current value of LVRCNT from common LVREGS

Subroutine LVSETP initializes all the continuants requested through array LVSTAK and upon completion brings page one, continuant zero, back into the buffer.

COMMON LVBUFR

Common LVBUFR contains all the variables for determining of the memory buffer size and the continuant locations on disk. The order in which the variables are listed here does not necessarily match the actual order as shown in Appendix A. The internal names for each variable are noted in brackets.

LVVSZE - Single array length[†] of the continuant

[PAGSZE] = (n*64) -2 where $1 \leq n \leq 8$

LVHDRS - Single array length of the header

[HDRSZE] = 2

LVPGHD - Single array length of the combined continuant and header

[PAGHDR] = PAGSZE + HDRSZE = n*64 where $1 \leq n \leq 8$

[†] The four arrays--NODSPC, LSTSPC, LNKSPC, and FLGSPC--are of equal length.

LVBKSZ - Number of blocks (256 words each) required to hold one continuant on disk
[BLKSZE] = 4 * PAGHDR/256

LVPGH4 - Total length, in words, of one continuant
[PAGHD4] = 4 * PAGHDR

LVNCOR - Number of continuants which reside in the in-core buffer [INCORE]

LVDRSZ - Single array length of the in-core directory. It must be a multiple of 64
[DIRSZE] = 64 * ((INCORE/64) + 1)

LVBFSZ - Total single array length of the in-core buffer. NODSPC, LSTSPC, LNKSPC, and FLGSPC are all dimensioned to this value
[BUFSZE] = DIRSZE + (INCORE * PAGHDR)

LVDRBK - Number of blocks required to hold the in-core directory on disk
[DIRBLK] = 4 * DIRSZE/256

LVMSAD - Location on disk (relative block number) of the in-core directory
[MSADIR] = 2

REPRESENTATION OF NODES AND LINKS

Before nodes and links may be used in a graph, they must be assigned to a page and given a random number which is unique to that page. This is accomplished by calling subroutine LVGRN. Page numbers can be either specifically requested by the user (1 ≤ LVREQP(1) ≤ 63) or assigned by LVGRN (LVREQP(1) = -1) to a new page. The random number returned is in the range of 1 to LVVSZE (the continuant size) and the same sequence of random numbers is repeated for each page. The total number of nodes and links which the user may define for any one page may not exceed LVVSZE or the program will terminate. Unless the default values are modified, nodes and links have the following form:

15 10	9 0
Page No.	Random Number

PAGE AND CONTINUANT DETERMINATION FOR THE TRIPLE

Page Determination

Subroutine LVINEX determines the page (and continuant) on which a triple is placed. The information needed for page placement is extracted from the source node at the time of insertion. If the source node is fully defined (from subroutine LVGRN), the prefix determines page placement. If the source node is not fully defined, it is expected to have one of the following values:

-1,	Place the triple on a new page
0,	Place the triple on the current page
$1 \le n \le 63$,	Place the triple on page n

In all these cases, LVINEX will call LVGRN to fully define the source node.

A request to place the triple on a new page is a special case. Two variables are used to compute a page number. LVHAPG, from common LVREGS, is an internal counter which keeps track of the highest page number in which there has been an insertion or for which a random number has been generated. LVHREQ, also from common LVREGS, is set by the user at the beginning of the program. This variable defines the number of pages created prior to execution of the program. During the course of execution, if LVHAPG(1) exceeds LVHREQ, continuant zero of a new page is created and LVHAPG(1) is incremented by one.

Continuant Determination

Before an insertion, deletion, or retrieval may take place, the particular continuant must be determined. If the user does not specify a continuant, all the continuants of the requested page will be examined in sequential order until either the requested function is found or the set of continuants for that page is exhausted. If the function does not exist, the triple is placed on the (sequentially) first continuant of the requested page which has available space. The continuant request is made with variable LVREQP(2) from labeled common LVREGS, which may take the following values:

$0 \le n \le 63$,	Continuant n is requested
-1,	New continuant is requested
-2,	Continuant is unspecified (default)
-3,	Current continuant if requested page is current page

If a value is to be added to a list that has been specifically placed on a particular continuant, but a different continuant is specifically requested, subroutine LVREOR reports an error. However, the insertion proceeds with the entire list moved onto the newly requested continuant.

With judicious use of subroutine LVREOR, two continuants may be MERGEd[2] and also a list may be SEPARATEd[2] from one continuant and placed on another.

THE FLAG FIELD

The flag field, contained in FLGSPC in Common LVVTR4, consists of eleven one-bit flags and two two-bit flags:

FLGSPC

14	13	12	11	10	9-8	0	1	2	3	4	5	6-7

Each flag describes a different aspect of the contents of the associated location in the buffer (Table 2). The first seven flags are the same as those for in-core GIRS.

TABLE 2 - THE FLAG FIELD

FLAG	FLAG VALUE	CONTENTS OF ASSOCIATED LOCATION
Flag 0	2^7	Head of a multivalued list.
Flag 1	2^6	Location already occupied.
Flag 2	2^5	A value on a multivalued list.
Flag 3	2^4	A node or link value. Does not refer to the actual contents of the location. Rather, the location value itself is used as a random number to define either a node or a link.
Flag 4	2^3	Head of a multivalued list which has been modified either by an insertion or by an indexed deletion, thus bypassing the "saved index" upon retrieval feature. (See the description of Subroutine LVFNV for further details.)
Flag 5	2^2	Head of a conflict list.
Flag 6-7	$2^1 + 2^0$	Type of value contained in the location: 00 Random number 01 Numeric data 10 Continuing string of Hollerith data 11 The only, or final, cell in a Hollerith data string
Flag 8-9	$2^8 + 2^9$	Type of triple contained in the location: 00 NODE LINK value 01 NODE value NODE 10 value LINK NODE
Flag 10	2^{10}	MVL backward continuation flag. This continuant does not contain the beginning of the list. A portion of this function resides on a lower-numbered continuant.
Flag 11	2^{11}	List forward continuation flag. This continuant does not contain the end of the list. A portion of this function resides on a higher-numbered continuant.
Flag 12	2^{12}	Inhibit reorganization of this list onto another continuant.
Flag 13	2^{13}	Head of a list which is a non-movable continuation of a list on some other continuant.
Flag 14	2^{14}	Pointer to sequence space.

DISK FORMAT

A saved GIRS file contains (in sequence) the following information: System values from the labeled commons, up to 228 user identifiers from labeled common LVUSER, variables for generating or continuing the random number sequence for up to 64 different pages, a copy of the directory of the continuants residing in the buffer when the program terminated, a directory containing the disk locations (relative to the beginning of the file) of all the continuants in the system, and copies of all of the continuants in the system.

The continuants are sequentially placed onto the file in the order of their creation. At the beginning of a "creation" type program, empty copies of all requested continuants are placed onto the file in sequence of increasing pages and continuants. After that, continuants are placed onto the file as they are created.

Also, at the beginning of a creation type program, sixteen blocks are allocated for the "out-core" directory. Each block holds the relative locations for the continuants of four pages. Otherwise, the block contains zeros.

Table 3 describes the disk format for a GIRS file which has been saved.

TABLE 3 - THE DISK FORMAT

Relative Location (in blocks)†	Size (in blocks)†	Contents
0	1	GIRS system variables from labeled commons LVREGS, LVRAND, LVBUFR, and LVVSEQ. Also, up to 228 user identifiers from labeled common LVUSER.
1	1	LVNTBL (256) from labeled common LVRAND.
2	LVDRBK	Directory of continuants residing in the buffer. LVDRBK = ((LVNCOR/64)+1)/64)
LVDRBK+2 • • •	1	Directory containing the locations (relative to the beginning of the file) of all continuants from pages 1-4.
LVDRBK+2+n	1	Out-core directory for all continuants from pages n*4+1 to n*4+4 where 0≤n≤15.
LVDRBK+18	LVBKSZ	Page 1, Continuant 0 LVBKSZ = LVVSZE/64 (the continuant size, LVVSZE, must be a multiple of 64)
LVDRBK+18 +LVBKSZ	LVDKSZ	Page 1, Continuant 1 or Page 2, Continuant 0; continuants are placed sequentially as they are created.
LVDRBK+18 +n*LVBKSZ	LVDKSZ	nth continuant to be placed onto the file.

†Each block contains 256 words.

PAGING SCHEME

GENERAL DISCUSSION

All the general I/O for out-core GIRS is handled on the PDP-11 computer by two RT-11 System Subroutine Library routinec: IREADW and IWRITW. These two routines operate in a block-oriented, random access, unformatted mode. They are called by four GIRS routines: LVPAGR and LVPAGW, to read in and write out the continuants; and LVDRRD and LVDRWR, to read in and write out any of the sixteen directories of continuant locations on disk. The I/O channels are initialized when GIRS subroutine LVSETP calls RT-11 System Subroutine Library functions: ICSI, IGETC, IFETCH, IENTER, and LOOKUP. The new channel is closed when GIRS subroutine LVDUMP calls subroutine LVCLOS which in turn calls System Subroutine Library routine CLOSEC. Since only six GIRS subroutines interact with the RT-11 System Subroutine Library, the I/O functions are relatively isolated. This leaves an otherwise portable all FORTRAN package.

I/O FOR THE DIRECTORIES OF DISK LOCATIONS OF CONTINUANTS

There are sixteen out-core directories with enough space to locate up to 64 continuants for each of 64 pages[†] on disk. Each directory has 256 words (one block) to locate the continuants for four consecutive pages: 1-4, 5-8, . . . etc. The directories are located on relative locations LVDRBK+2 through LVDRBK+17 of the disk file. Only one directory at a time is stored in main memory in array LVOTDR(256) in labeled common LVREGS. To find the desired location within the directory three variables also from labeled common LVREGS are needed: LVDRPG, LVDIRC, and LVOTLC. LVDRPG contains the current directory number as determined by the last requested page.

$$\text{LVDRPG} = (\text{Page No.} -1)/4 + 1$$

$$(\text{value range} = 1\text{-}16)$$

LVDIRC determines the quadrant number within the directory for the requested page.

$$\text{LVDIRC} = \text{Page No.} - 4*(\text{LVDRPG} -1)$$

$$(\text{value range} = 1\text{-}4)$$

LVOTLC is the position within the directory of the disk location for the requested page and continuant number.

$$\text{LVOTLC} = 1 + 64*(\text{LVDIRC} -1) + \text{Cont. No.}$$

$$(\text{value range} = 1\text{-}256)$$

[†] Practical considerations limit the number of pages to a maximum of 63, not 64.

I/O FOR THE CONTINUANTS

I/O for the continuants is controlled by an I/O executive routine, LVEXCH. LVEXCH takes as input a requested page and continuant number, LVREQP(1) and LVREQP(2) from labeled common LVREGS, and either confirms its current residency in the GIRS buffer or brings it into the buffer. In either case, the current page (LVCUPG(1)) and LVCUPG(2) from labeled common LVREGS) is updated to the requested page.

The general flow of LVEXCH is as follows:

1) Call LVDRCT to search the in-core directory and determine whether the requested page and continuant (REQ(P,C)) are in the buffer. If so, update the "current page" register and return.

2) Call LVMSA to bring into main memory the correct "Directory of Continuant Locations on Disk" if necessary and then determine whether the continuant exists and if so, its location on disk.

3) Call LVOPEN to make a control point ("continuant block") available in the GIRS buffer. If the buffer contains more than one continuant block, call LVALUE to determine (using the continuant header values) which in-core continuant is of least value. If the current continuant has been modified since it was brought into the buffer, write it out to disk. (The algorithm used for this determination is discussed in the next section.)

4) Call LVPAGR to bring the requested page into the GIRS buffer.

5) Update the "current-page" register (LVCUPG()).

6) Call LVRPLC to update the In-Core directory.

7) Call LVSUM to update the new continuant header and then return.

PHILOSOPHY

The philosophy used by Out-Core GIRS for bringing in continuants is generally known as "demand paging," that is, a continuant is brought into the buffer only when it is specifically requested. However, any continuant presently residing in the buffer must be saved before it is written over if it has been modified by an insertion or deletion. Furthermore, if the buffer holds more than one "continuant block," a specific continuant must be selected for removal.

Subroutine LVALUE contains the formula[†] used for this purpose. It is a modification of an optimization formula designed for the Control Data Corporation (CDC) Interactive Graphics Data Handler.[6]

Each continuant has a desirability value computed from values stored in the header of that continuant. The continuant with the lowest desirability value is either written over or written out to disk, of course. The formula is:

$$\text{value} = A * \text{order} + B * \text{usage} + C * \text{space} + D * \text{write}$$

where the weighting factors A, B, C, and D sum to 100. The weighting factors are set as follows: A = 15.0, B = 20.0, C = 15.0, and D = 50.0. Order is a measure of how long the continuant has been in core. Continuants most recently read in are weighted more heavily. Usage is the ratio of the use count for an individual continuant to the total usage for all the continuants in the buffer at the time of the computation. Usage is defined as the sum of all calls to subroutines LVINEX, LVFDEX, and LVDLEX which reference a particular continuant from the time that continuant was read into the buffer. Space refers to the fill ratio of the continuant. The emphasis of the fill ratio varies with the type of computer run. For a creation type run, a half-filled continuant is emphasized and for a production type run, a 5/8 to 7/8 filled continuant is emphasized and an empty continuant is deemphasized. The write parameter greatly emphasizes a continuant which has been modified because of the immediate 50 percent savings in disk I/O if the present continuant does not have to be written out to disk prior to reading in the requested continuant.

[†] This formula was devised by Mr. M. Haas, formerly with CDC and with DTNSRDC.

USER-CALLABLE GIRS SUBROUTINES

INITIALIZATION

Getting Started

To execute a GIRS program, the following labeled commons and declarations should be included in the driving program:

```
     REAL*4 DEFEXT,LVCORE
     LOGICAL*1 LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
                              LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
                              LVFD4,LVDL4,LVIN4,LVCRNT

COMMON /LVARGS/ LVFUNC,LVVARG,LVVPOS,LVVTYP,LVVAL,LVVNVL,LVSKIP,
1               LVVTR,LVVINC,LVNDXN,LVVALS(10),LVTYPE(10)
2               ,LVSRSF,LVLNSF,LVSNSF,LVNTYP
COMMON /LVVSEQ/ LVSIZE,LVSEQ1,LVSEQ2,SEQSPC(1)
COMMON /LVRAND/ LVKPRM,LVKS,LVKY,LVKDY,LVKDX,LVTEMP,LVLIST,LVNTBL(256)
COMMON /LVVTR1/ NODSPC(buffer size)
1      /LVVTR2/ LSTSPC(buffer size)
2      /LVVTR3/ LNKSPC(buffer size)
3      /LVVTR4/ FLGSPC(buffer size)
COMMON /LVCRNT/ LVVGSP,LVCTRL,LVCTR1,LVLSTV,LVNFRE,LVFREE,
1               LVDREG,LVVMSA,LVPGLC,LVCRNT
COMMON /LVBUFR/ LVVSZE,LVNWCH,LVOLCH,LVCMPR,LVPGHD,LVBFSZ
1               LVDRSZ,LVNCOR,LVHDRS,LVMSAD,
2               LVSFSZ,LVBKSZ,LVDRBK,LVPGH4
COMMON /LVREGS/ LVCUPG(4),LVREQP(4),LVLVPG(4),LVMSAR,
1               LVHRPG,LVNMSA,LVHAPG(2),LVRCNT,LVUCNT,LVDRPG,
2               LVDIRC,LVOTLC,LVOTDR(256),
3               LVRWBF(4*continuant size)
COMMON /LVPRAM/ LVBFLC,LVLNTH,LVVERR,LVERNO,LVBNRY,LVBCD,LVMODE,LVPGS,LVLUN
COMMON /LVRUN/  LVRNTP,LVCORE
COMMON /LVSTAK/ LVLEVL,LVNVAR,LVSTAK(140)
COMMON /LVMASK/ LVWRIT,LVNUSE,LVNWCN,LVMSK3,LVMSSF,LVMSPF
COMMON /LVSWIT/ LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1                  LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2                  LVFD4,LVDL4,LVIN4
COMMON /LVUSER/ USER(228)
COMMON /LVUTIL/ FILSPC(39),DEFEXT(2)
```

Note:

1) A user may place up to 228 identifiers in Common LVUSER. These identifiers will automatically be placed on disk if a file is created.

2) If the "swap USR" function* of the RT-11 operating system for the PDP-11 computer is on (default), then COMMON /LVUTIL/ should be placed at the end of the set of labeled commons to prevent its being swapped out of main memory. If this labeled common is swapped out of main memory, the operating system as well as the program will go down as soon as the input and output file names are read in. If this common block is placed at the end and the system still goes down, either "SET USR NOSWAP" or try placing a dummy array in front of the common.

The following declaration and labeled commons should be included in all subroutines in which there are GIRS operations:

```
LOGICAL*1 LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1                LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2                LVFD4,LVDL4,LVIN4,LVCRNT
COMMON /LVARGS/ LVFUNC,LVVARG,LVVPOS,LVVTYP,LVVAL,LVVNVL,LVSKIP,
1                LVVTR,LVVINC,LVNDXN,LVVALS(10),LVTYPE(10),
2                LVSRSF,LVLNSF,LVSNSF,LVNTYP
COMMON /LVREGS/ LVCUPG(4),LVREQP(4),LVLVPG(4),LVMSAR,
1                LVHRPG,LVNMSA,LVHAPG(2),LVRCNT,LVUCNT,LVDRPG,
2                LVDIRC,LVOTLC,LVOTDR(256),LVRWBF(512)
```

*The "swap USR" function will swap out of main memory the first 2000 words of a user's program in order to bring in RT-11 system routines.

```
COMMON /LVSWIT/ LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1               LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2               LVFD4,LVDL4,LVIN4
```

If Subroutine LVDUMP is to be called from a subroutine, the following labeled COMMON is needed:

```
COMMON /LVPRAM/ LVBFLC,LVLNTH,LVVERR,LVERNO,LVBNRY,LVBCD,
1               LVMODE,LVPGS,LVLUN
```

In order to initialize the GIRS buffer and the random number generator, LVSETP must be the first GIRS subroutine called. The following variables must also be defined prior to the call to LVSETP and any calls to LVGRN:

LVSTAK()
LVVSZE
LVHRPG
LVSFSZ[†]
LVSIZE
LVNCOR
LVMSPF[†]
LVKPRM
LVRNTP
LVMSSF[†]

These variables are described in subsequent sections on LVSETP and LVGRN.

The letters "LV" must not be used to begin subroutine and variable names. These initial letters are reserved for GIRS.

The user must first decide on a continuant size (LVVSZE), which determines the maximum number of nodes and links that may be defined for a given page. Its value must be (n*64)-2, n > 0.[††] Next, the user must decide how many continuants may be present in core simultaneously (LVNCOR). This value will determine the in-core directory size (LVDRSZ) as computed by LVSETP to be 64*((LVNCOR/64)+1). Consequently, the space needed for each of the four fields (NODSPC, LSTSPC, LNKSPC, and FLGSPC) of the GIRS buffer is

64*((LVNCOR/64)+1) + LVNCOR*(LVVSZE+2)

The user must then decide whether "Sequence Space" will be used. If so, LVSIZE is set to that value; otherwise, LVSIZE is set to 1. Also, the user must dimension array LVRWBF from Common LVREGS to:

4*(continuant size+2)

Note that this dimension must be a multiple of 256.

[†] These variables have default values and need be defined only if the node suffix and prefix sizes are modified.

[††] When used with GIRL, it must be a multiple of 64.

GIRS expects a program to do one of the following things:

1) create a new graph

2) update an old graph

3) query an old graph

LVRNTP must be set to one, two, or three to indicate the type of program to be executed.

It is more efficient for pages (and continuants of those pages) to be initialized at the beginning of execution of a program than to be created on demand. Set LVHRPG to the highest page number desired. There is a limit of 63 pages unless LVSFSZ is modified. LVSFSZ is the node suffix size and has a default size of ten bits, which allows for a maximum continuant size of $2^{10}-1$ or 1024. The prefix size is therefore six bits which allows for $2^{6}-1$ or 63 pages. Changing the suffix size will modify these upper limits accordingly. If LVSFSZ is modified, the prefix and suffix masks, named LVMSPF and LVMSSF, must be updated accordingly. For example, if LVSFSZ is set to 12, set

LVMSPF to 170000

and LVMSSF to 7777

Continuants for each page may be initialized as follows:

Set the I^{th} location in array LVSTAK() to the number of continuants desired (beyond the zeroth) for page I. There is a limit of 63 extra continuants per page. Set the rest of the 140 locations in LVSTAK() to zero.

To initialize the random number generator (LVGRN), set LVKPRM to the first prime number $\geq(\sqrt{\text{LVVSZE}})/2$.

Finally, before calling LVSETP, an output file to contain error statements should be assigned a logical unit number (LVLUN). The following statements, for example, will work on the RT-11 operating system for the PDP-11 computer:

```
LVLUN = 17
CALL ASSIGN(17,'SY:ERROR.ERR',12)
```

When an identifier is defined by the random number generator (LVGRN), it is given a prefix (a page definition) and a suffix (a random number, unique to that page). The user must assign the prefix via LVREQP(1). The value range for LVREQP(1) is 1 to 63.

Note that if LVDUMP is called, up to 228 variables may be automatically saved at the end of a program if they are placed into COMMON /LVUSER/.

Subroutine LVSETP

Function:

Initializes the I/O channels for the files containing the old and new graphs. Initializes those variables needed for Subroutine LVGRN. Initializes the in-core and out-core directories. Initializes all requested continuants and places them onto disk.

Calling Format:

CALL LVSETP

Input Parameters:

(In COMMON /LVREGS/)

LVHRPG   Highest initially requested page. No default.

(In COMMON /LVRAND/)

LVKPRM   First prime number $\geq(\sqrt{LVVSZE})/2$.

(In COMMON /LVBUFR/)

LVVSZE   Continuant size; similar to MEMSZE from in-core GIRS. No default.

LVNCOR   Number of continuant slots in the in-core GIRS buffer. No default.

LVSFSZ   Node suffix size, default is ten bits.

(In COMMON /LVRUN/)

LVRNTP   Type of run:

= 1 Create a new graph (default)

= 2 Update an old graph

= 3 Query an old graph

Comments:

LVSETP must be the first GIRS subroutine called by the driving program since it is the main initialization routine. Before initializing the GIRS buffer and other tables, the user is prompted at the teletype for file names for the old and new graphs. The user response must include both names, in command string format, even if only one is needed. The default extension for both file names is .GRF.

LVSETP is in overlay region 1, segment "SETPOP".

Program Length:

2615_8 1421_{10}

Subroutines Called:

LVFECH
LVGRN
LVPAGW
LVNSRT
LVDRWR
LVMSA
LVFIND
LVPAGR
ICSI
IGETC
IFETCH
IENTER
LOOKUP

Called by the Following Subroutines:

LVNPAG
LVNCON

Subroutine LVGRN

Function:

Assigns a page and "random number," which is unique to that page, to a given GIRS identifier.

Calling Format:

CALL LVGRN(NODE)

Input Parameters:

(In COMMON /LVREGS/)

LVREQP(1)	Requested page number
	= 0, define an identifier on a new page.
	$1 \leq n \leq 63$, define an identifier on page n.

(In COMMON /LVRAND/)

LVLIST	Number of identifiers to be assigned random numbers. Default is one.

Output Parameters:

(Format Argument)

Node	*Contains* generated random number. It must be dimensioned to "LVLIST" if LVLIST > 1.

Comments:

For each page, a repeatable sequence of unique random numbers is generated in the range of 1 to LVVSZE. LVLIST numbers are generated per call. An attempt to define more than LVVSZE number of identifiers for any one page will terminate the program unless a random number has been "undefined" by Subroutine LVRTRN. Identifiers must be integers. The generated sequence has been previously described by Berkowitz[2] and Zaritsky.[1]

Equivalent GIRS Code:

Identifiers may be defined in GIRL in at least two ways. At the beginning of each routine, a list of identifiers may be defined for page n in the following manner:

```
G DEFINEn NODE1,..., NODEk
```

Identifiers may be given random numbers at any time with the following code:

```
  LVREQP(1) = "page number"
G $'NODEi
```

Identifiers may also be defined during the execution of an insertion, as discussed further under "Insertion."

Program Length:

750_8 488_{10}

Subroutines Called:

LVLFSH

LVEXCH

LVERR

Called by the Following Subroutines:

LVSETP

LVINEX

RETRIEVAL OF VALUES

Discussion

Value retrieval is overseen by the find executive routine LVFDEX. This routine brings in the proper continuant so that the lower level routines LVFIND and LVFNV may search for the desired function and value. If the continuant is not specified (default), all the continuants of the requested page will be searched in sequential order until either the function is found or all the continuants have been examined. If the continuant has been specified and the search is to be from "top-to-bottom," the search will proceed to the next higher numbered continuant only if FLAG-11 has been set for that list. If the continuant has been specified and the search is to be from "bottom-to-top," the search will proceed to the previous (lower-numbered) continuant only if FLAG-10 has been set for that list.

LVFDEX expects the user to provide two find strategy routines: USRFD1 and USRFD2. USRFD1 precedes the actual retrieval, but it is skipped if LVFD1 is .FALSE. (default). The retrieval may be skipped if LVFD4 is set within USRFD1 to .FALSE. (default is .TRUE.). USRFD2 follows the retrieval but it is skipped if LVFD2 is .FALSE. (default). If USRFD1 is called, LVFD2 may be modified by LVFD3. USRFD1 and USRFD2 cannot be used recursively.

Subroutine LVFDEX

Function:

a) Calls user find strategies USRFD1 and USRFD2, skipping the retrieval if LVFD4 is .FALSE.

b) Brings in the proper continuant or sequence of continuants (if there is no specific request) in preparation for the retrieval.

c) Breaks up LVFUNC and LVARG into their prefix (page) and suffix (random number) components.

d) Oversees the following operations:

1) function address computation.
2) determination of function existence. If the function does exist, then
3) location of function within the continuant (since it may not be first on the conflict list, and may therefore reside anywhere in the continuant).
4) determination of whether the function is an SVL or MVL.
5) location in continuant of preceding function on the conflict list.
6) retrieval of the IPOSth value (and its location) of the type indicated, from the top or bottom (depending on the sign of IPOS) of a list of values of a specified function.

Calling Format:

Call LVFDEX(INDEX,INDXAD,KFUNC,KARG,SAVCON)

Input Parameters:

(In COMMON /LVARGS/)

LVFUNC Link of the triple, also known as the function. The value in IFUNC must contain both a prefix (page number) and a suffix (random number) as defined by a call to LVGRN.

LVVARG Source node of the triple, also known as the argument of the function. The value in IARG must contain both a prefix (page number) and a suffix (random number) as defined by

LVGRN. The source node prefix determines the page placement of the function and hence the page on which to search.

LVVPOS Position in the multivalue list, IPOS locations from the top (if IPOS is positive) or from the bottom (if IPOS is negative). If ITYP is specified, only that type of value is considered in determining the position.

LVVTYP Type of value to be retrieved:

= 0 Random number plus page

= 1 Integer data

= 2 Hollerith data

= 3 No specified type (default value)

LVSKIP Saved-index defeat switch. If LVSKIP = 1, the saved-index operation is skipped; otherwise the saved-index feature is in effect. LVSKIP can be set either at the start of the program or just before a call to LVFDEX (after which it may be reset). The saved-index feature is described by Zaritsky.[1]

(In COMMON /LVREGS/)

LVREQP(2) Requested continuant:

= -2 continuant unspecified

$0 \leq n \leq 63$, continuant n is requested

(In COMMON /LVSWIT/)

LVFD1 = .TRUE. call user's first retrieval strategy routine.

= .FALSE. skip user's first retrieval strategy routine (default)

LVFD2 = .TRUE. call user's second retrieval strategy routine.

= .FALSE. skip user's second retrieval strategy routine (default)

[Input from USRFD1]

LVFD3 may be used to modify LVFD2

LVFD4 = .TRUE. proceed with the retrieval (default)

= .FALSE. skip the retrieval

(Formal Parameter Set)

The formal parameter set is needed by LVFDEX when the saved-index option is to be used. The parameter set consists of five variables, each of which must be unique for each new call to LVFDEX involving a saved index.

KARG   Source node associated with a particular call to LVFDEX

KFUNC   Link associated with a particular call to LVFDEX

INDEX   Position in the list of the value retrieved from the most recent call to LVFDEX. If INDEX is negative, it is the position from the bottom of the list.

INDXAD   Location in continuant SAVCON of the value retrieved from the most recent call to LVFDEX

SAVCON   Continuant on which list resides

Output Parameters:

(In COMMON /LVARGS/)

LVVPOS   Set to 1 (default value)

LVVTYP   Set to 3 (default value)

LVVAL   Retrieved value (LVVPOSth value of the type LVVTYP). LVVAL is set to LVVARG if the value cannot be found.

LVVTR   If the LVVPOSth value of the LVVTYP exists, LVVTR = 1; otherwise LVVTR = -1.

(In COMMON /LVREGS/)

LVCUPG(1) Current page. If LVVTR=1, LVCUPG(1) is set to the page containing the requested function.

LVCUPG(2) Current continuant. If LVVTR=1, LVCUPG(2) is set to the particular continuant containing the requested function.

Equivalent GIRL Code:

```
C SET LVREQP(2) TO THE REQUESTED CONTINUANT, IF DESIRED.
G NODE+LINK.tsJ
```

where t is the type of value to be retrieved:

= "	Identifier (node defined by LVGRN)
= .	Integer value
= /	Hollerith value
= "blank"	Any type value

s is the indicating direction of search:

= + or

"blank"   Search from top of list

= -   Search from bottom of list

J is the same as IPOS (=LVVPOS)

<u>Program Length</u>:

1254_8 686_{10}

<u>Subroutines Called</u>:

LVSTAC   LVFIND   LVRTSH   LVBOTM

LVPOP   LVFNV   LVEXCH

<u>Called by the Following Subroutines</u>:

LVDLEX   LVINEX   LVINCL

RETRIEVAL OF MVL INDEX OF GIVEN VALUE OF A FUNCTION (INCLUSION)

Subroutine LVINCL

Function:

Determine the first MVL position of a given value.

Calling Format:

CALL LVINCL

Input Parameters:

(In COMMON /LVARGS/)

LVVINC Value on which the list position is to be determined

Output Parameters:

(In COMMON /LVARGS/)

LVVPOS First position in the MVL in which the indicated value is found

LVVINC = 1 Desired value has been found on the MVL
 = -1 Desired value has not been found on the MVL

LVVTR Same as LVVINC

Equivalent GIRL Code:

Use of the GIRL inclusion operator can best be explained with three examples. Further discussions and examples are given in Berkowitz.[5]

Assume for all examples that the source node is NODE and the link is LINK:

Example 1. Delete value3 on the MVL

```
G NODE+LINK-.:value3
```

Example 2. Determine the position of valuei (if such a value exists) on the MVL and name it INDEX; otherwise transfer to fail.

```
G NODE+LINK   valuei/fail′:INDEX
```

Example 3. Replace valuel on the MVL with value2.

```
G NODE LINK: valuel value2
```

Program Length:

230_8 152_{10}

Subroutine Called:

LVFDEX

:NSERTION

Discussion

The insertion operation is overseen by the Insert Executive Routine LVINEX. This routine ensures that the tripl. is completely defined and then determines placement of that triple. If the triple is already fully defined, the requested page is determined by the prefix of the source node (LVVARG). It is in the domain of insertion that a new page or continuant can be requested. If a particular continuant is not requested and the function did not previously exist, the triple is placed on the (sequentially) first continuant with available space. If a value is to be added to a list that has been specifically placed on a particular continuant, but a different continuant is specifically requested, subroutine LVREOR reports an error. The insertion proceeds, however, with the entire list moved onto the newly requested continuant.

LVINEX expects the user to provide two insertion strategy routines: USRIN1 and USRIN2. USRIN1 precedes the actual insertion, but it is skipped if LVIN1 is .FALSE. (default). The insertion may be skipped if LVIN4 is set within USRIN1 to .FALSE. (default is .TRUE.). USRIN2 follows the insertion, but it is skipped if LVIN2 is .FALSE. (default). If USRIN1 is called, LVIN2 may be modified by LVIN3. USRIN1 and USRIN2 cannot be used recursively.

Subroutine LVINEX

Function:

a) Calls user insertion strategies USRIN1 and USRIN2, skipping the insertion if LVIN4 is .FALSE.

b) Ensures that the source node, link and, if a random number, the sink node are all completely defined (contain both a prefix and suffix).

c) Determines on which page and continuant to place the triple and brings that continuant into the buffer, if necessary.

d) Oversees the actual insertion by Subroutine LVNSRT.

Calling Format:

CALL LVINEX

Input Parameters:

(In COMMON /LVARGS/)

LVFUNC — Link of the triple, also known as the function. A fully defined link contains a prefix* (page number) and a suffix (random number) as given by LVGRN.

$= 0$	Define the link with a prefix set to the current page
$= 1 \leq n \leq 63$	Define the link with a prefix set to n

LVVARG — Source node of the triple, also known as the argument of the function. The prefix of LVVARG determines on which page to place the triple. A fully defined node contains a prefix and

*Unless modified by the user, the prefix consists of the leftmost six bits of the node or link.

a suffix as given by LVGRN. Otherwise,

= -1 Place triple on a new page and define the node

= 0 Place triple on the current page and define the node

$1 \le n \le 63$ Place triple on page n and define the node

LVVNVL Number of values (up to ten) to be inserted (default is 1)

LVTYPE(10) Type of each value in LVVALS(i) to be inserted:

= 0 Random number (default value)

= 1 Integer data

= 2 Continuing Hollerith data

= 3 The only or final cell or a Hollerith data string

LVVALS(10) Array containing the values or sink nodes to be inserted. LVVALS(i) may contain any of the following types of values:

- Random number, as defined by LVGRN
- Integer data; see Berkowitz[2] for limitations on Integer data
- Hollerith data; see Berkowitz[2] for limitations on Hollerith data

If LVTYPE(i) = 0 (random number), LVVALS(i) may also take on the following forms:

= -1 Define the sink node with a prefix = "current page + 1"

= 0 Define the sink node with a prefix set to the current page

$1 \le n \le 63$ Define the sink node with a prefix set to n

LVNTYP Orientation of insertion

= 0 Insert sink node (default)

= 1 Insert source node

= 2 Insert link

LVNDXN Type of insertion to be made:

= 0 Normal insertion; the triple is always placed at the end of the (null) list. This is the default value.

= 1 Destructive insertion; the contents of the LVVPOSth member of the LVVTYP type (counting from the top or bottom of the list, depending on the sign of LVVPOS) are replaced by the contents of LVVALS(1).

= 2 Nondestructive insertion; the contents of LVVALS(1) are wedged into the list, making the new value the LVVPOSth member of the LVVTYP type from the top or bottom of the list (depending on the sign of LVVPOS).

The following two variables are needed only if LVNDXN = 1 or 2:

LVVPOS LVNSRT will place the value to be inserted LVVPOS locations (as modified by LVVTYP) from the beginning or, if negative, from the end of the list.

LVVTYP Type of value to be counted when attempting to insert a value at LVVPOS locations from the beginning or end of a list.

(In COMMON /LVREGS/)

LVREQP(2) Requested continuant. Note that LVREQP(1) contains the requested page which is extracted from LVVARG.

= -1 Request new continuant

= -2 Continuant unspecified (default)

= -3 Current continuant (if current page = requested page)

$0 \le n \le 63$ Continuant n is requested

(In COMMON /LVSWIT/)

LVIN1 = .TRUE. Call user's first insertion strategy routine

= .FALSE. Skip user's first insertion strategy routine (default)

LVIN2 = .TRUE. Call user's second insertion strategy routine

= .FALSE. Skip user's second insertion strategy routine (default)

(input from USRIN1)

LVIN3 May be used to modify LVIN2

LVIN4 = .TRUE. Proceed with the insertion (default)

= .FALSE. Skip the insertion

Output Variables:

(In COMMON /LVARGS/)

LVVPOS Set internally to 1 (default value)

LVVTYP Set internally to 3 (default value)

LVVAL Set internally to LVVALS(1)

LVVNVL Set internally to 1 (default value)

LVVTR = -1 Function did not exist prior to this insertion

= 1 Function did exist prior to this insertion

LVNDXN Set internally to 0 (default value)

(In COMMON /LVREGS/)

LVCUPG(1) Current page (as a result of this insertion)

LVCUPG(2) Current continuant of current page. (Contains inserted triple.)

Equivalent GIRL Code:

Assume that NODE1 is the source node and LINK1 is the LINK and (in the first four examples) both NODE1 and LINK1 have been initialized in a DEFINEn statement:

1) Add random number valuei to the (null) list:

```
G    NODE1    LINK1    valuei
```

2) Add integer I to the end of the list

```
G    NODE1    LINK1    "I"
```

3) Place valuei in the third location from the bottom of the list.

```
G      NODE1    LINK1    .-3    valuei
```

4) Replace the second integer value from the top of the list with the integer 10.

```
G      NODE1    LINK1-..2    "10"
```

5) Assign a random number to valuei (for the current page) and place the triple on page 5, continuant 0.

```
          NODE1 = 5
          LVREQP(2) = 0
          VALUEI = 0
    G     NODE1    LINK1    VALUEI
          PRINT NODE1, VALUEI
```

6) Place each of the following ten triples on new pages, assign random numbers to the source nodes, links, and sink nodes, and define each of the links and sink nodes to the page which is current at the time of definition of the source node. The triples will automatically be placed on the zeroth continuants of each new page.

```
             DO 5 I = 1, 10
             NODE = -1
             LINK = 0
             SINK = 0
    G        NODE LINK SINK
             PRINT NODE, LINK, SINK
    5        CONTINUE
```

7) Define NODE1, LINK1, SINK to page 3, place this triple on continuant 2, and call the first insert strategy routine.

```
    G      DEFINE3 NODE1, LINK1, SINK
                       .
                       .
                       .
           LVIN1 = .TRUE.
           LVREQP(2) = 2
    G      NODE1 LINK1 SINK
```

<u>Program Length</u>:

1622_8 914_{10}

Subroutines Called:

LVSTAC
LVPOP
LVNPAG
LVGRN
LVNCON
LVREOR
LVOVER
LVLFSH
LVRTSH
LVERR
LVEXCH
LVFDEX
LVFIND

Subroutine LVREOR

Function:

To move a list from its present location to a new continuant as specified by REQCON.

Calling Format:

```
CALL LVREOR(REQCON)
```

Input Parameters:

(Formal Parameter Set)

REQCON The list is to be moved from continuant "LVREQP(2)" to continuant "REQCON" of page "LVREQP(1).

Comments:

Subroutine LVFDEX must be called immediately prior to a call to this routine. If the original list was specifically placed on LVREQP(2), an error message is written out. Two continuants may be MERGEd together by calling LVFDEX and this routine once for each function in the original continuant (LVREQP(2)). A particular list may be SEPARATEd from one continuant and placed on another (REQCON) in the same fashion. The present version of LVREOR expects a new triple to be added to continuant REQCON each time it is called. Also, this triple must be fully defined.

DELETION

Discussion

The delete operation is overseen by the delete executive Routine LVDLEX. If no continuant is requested, LVDLEX brings in (sequentially) all continuants of the requested page (as defined by the prefix of the source node) until either the function is located or there are no more continuants of the requested page.

LVDLEX expects the user to provide two deletion strategy routines: USRDL1 and USRDL2. USRDL1 precedes the actual insertion, but it is skipped if LVDL1 is .FALSE. (default). The deletion may be skipped if LVDL4 is set within USRDL1 to .FALSE. (default is .TRUE.). USRDL2 follows the deletion but it is skipped if LVDL2 is .FALSE. (default). If USRDL1 is called, LVDL2 may be modified by LVDL3. USRDL1 and USRDL2 cannot be used recursively.

Subroutine LVDLEX

Function:

a) Calls user deletion strategies USRDL1 and USRDL2, skipping the deletion if LVDL4 is .FALSE.

b) Searches in sequential order (unless the continuant is specified) the continuants of the requested page for the requested function.

c) Oversees the actual deletion by Subroutine LVDLET.

Calling Format:

CALL LVDLEX

Input Parameters:

(In COMMON /LVARGS/)

LVFUNC Link of the triple; must be a random number as defined by LVGRN.

LVVARG Source node of the triple; must be a random number as defined by LVGRN.

LVNDXN = 0 Delete entire function (default)
= 1 Delete specific value as described by LVVPOS and LVVTYP

The following two variables are needed only if LVNDXN = 1:

LVVPOS Position in the MVL of the value to be deleted (number of locations from the top, if positive, and from the bottom, if negative). If LVVTYP is specified, only that type of value is counted in determining the position in the list. LVVPOS is used only for indexed deletion.

LVVTYP Type of value to be deleted from a multivalued list (used only for indexed deletion)
= 0 Random number
= 1 Integer data
= 2 Hollerith data
= 3 No specified type (default value)

(In COMMON /LVREGS/)

LVREQP(2) Requested continuant
= -2 Continuant unspecified (default)
$0 \le n \le 63$ Continuant n is requested

(In COMMON /LVSWIT/)

LVDL2 = .TRUE. Call user's first deletion strategy routine

= .FALSE. Skip user's first deletion strategy routine (default)

LVDL2 = .TRUE. Call user's second deletion strategy routine

= .FALSE. Skip user's second deletion strategy routine (default)

(input from USRDL1)

LVDL3 May be used to modify LVDL2

LVDL4 = .TRUE. Proceed with the deletion (default)

= .FALSE. Skip the deletion

<u>Output Parameters</u>:

(In COMMON /LVARGS/)

LVVAL Deleted value. If the entire list is deleted, LVVAL returns the first value of the list.

LVVTR Function indicator. If the function or specified value of that function does not exist, the attempted deletion is considered to have failed. LVVTR is actually set in LVFIND and LVFNV.

= 1 Function exists

= -1 Function does not exist

LVVPOS Set internally to 1 (default value)

LVVTYP Set internally to 3 (default value)

LVNDXN Set internally to 0 (default value)

(In COMMON /LVREGS/)

LVCUPG(1) Current page (contained deleted triple)

LVCUPG(2) Current continuant of current page

<u>Equivalent GIRL Code</u>:

Assume NODE1 is the source node and LINK1 is the link.

Example 1.

Delete entire function which begins on continuant 2.

```
        LVREQP(2) = 2
G       NODE1-LINK1
```

Example 2.

Delete the Ith value on an MVL, continuant is not known.

```
G    NODE1+LINK1-.I
```

__Program Length__:

762_8 498_{10}

__Subroutines Called__:

LVSTAC

LVPOP

LVERR

LVFDEX

LVDLET

USRDL1

USRDL2

DISK STORAGE AND RETRIEVAL OF A GRAPH

Discussion

After a graph has been created, it may be conveniently stored in binary format on disk and later retrieved from disk via the Subroutines LVDUMP and LVFECH. Although this task can be performed without these routines, their use ensures that all pertinent variables will be properly defined. LVDUMP also enables the user to have the entire graph, a single page of that graph, or the contents of the buffer generated in ASCII format for debugging purposes. The dump is placed on logical unit LVLUN which must be defined in a call to SYSLIB function ASSIGN.

Another advantage of this arrangement is that it makes it easy for the user to restart a program using new data. The original graph will be retrieved whenever a new call to LVFECH is made. If LVDUMP is called, up to 228 identifiers may be automatically saved at the end of a program if they are placed into COMMON /LVUSER/.

The names for the files containing the old and new graphs are declared at the beginning of execution of the program. A prompt character is sent to the teletype and the user response must include names, in command string format, for both an old and new graphs, even if only one is needed. The default extension for both file names is .GRF.

Subroutine LVDUMP

Function:

LVDUMP will either:

a) Store the entire graph, pertinent GIRS system variables, and 228 identifiers from COMMON /LVUSER/ onto the output file in a format suitable for later recovery by Subroutine LVFECH, or

b) For debugging purposes, create an ASCII file on logical unit LVLUN consisting of GIRS system variables plus one of the following:

1) The entire graph

2) A single page of the graph

3) Those continuants residing in the buffer at the time of the call to LVDUMP

Calling Format:

CALL LVDUMP(DUMP)

Input Parameters:

(In COMMON /LVFRAM/)

LVMODE Determines whether to invoke function "a" or "b"

= LVBNRY function "a"

= LVBCD function "b"

If function "b" is invoked, the following three parameters are needed:

LVPGS = -1 Output those continuants residing in the buffer at the time of the call to LVDUMP

= 0 Output all continuants of all pages

$1 \leq n \leq 63$ Output all continuants of page n

LVLUN Logical unit number of the ASCII file which will contain the output from LVDUMP. It must be defined in a CALL ASSIGN statement.

(Formal Parameter)

DUMP = 0 Output to LVLUN some of the pertinent GIRS variables found in the labeled commons

= 1 Output to LVLUN all the pertinent GIRS variables found in the labeled commons

Program Length:

553_8 363_{10}

Subroutines Called:

LVERR

LVWRIT

LVEXEC

LVPAGW

LVCLOS

Subroutine LVFECH

Function:

Reads in (in binary format) pertinent GIRS system variables, up to 228 user identifiers from labeled COMMON /LVUSER/, and a previously created graph from disk into the GIRS buffer. Then it copies the graph onto a new disk file.

Calling Format:

CALL LVFECH

Comments:

LVFECH expects the disk file to have been created by LVDUMP. At the beginning of the program LVFECH is called by LVSETP if LVRNTP = 1 or 2. LVFECH may be called by the user directly if there is a need to reinitialize the graph.

Program Length:

1036_8 542_{10}

Subroutines Called:

LVERR
LVPAGR
LVMSA
LVDRRD
LVDRWR
LVPAGW

Called by the Following Subroutine:

LVSETP

EXECUTING A GIRS PROGRAM

GENERAL DISCUSSION

GIRS may be used directly via user calls to the GIRS subroutines or indirectly with the GIRL[5] language. In either case, the object code for the driving program must precede the object code for the GIRS routines in any LINK-LOAD.

It is generally more advantageous for the user to use GIRS indirectly via GIRL, since GIRL not only includes all the capabilities of GIRS but also spares the user from concern over setting up all the labeled commons and initializing pertinent variables. The command sequences and FORTRAN statements needed to preprocess, compile, link, and execute GIRL/GIRS programs on the PDP-11 follow.

INDIRECT USE OF A GIRS SUBROUTINE VIA GIRL

A GIRL program must include the following statements:

```
 Options card
 Continuant specification card 1
 Continuant specification card 2 (if >25 pages specified)
 Continuant specification card 3 (if >50 pages specified)
 First user program card
     or
 $  SUBROUTINE name
    non-DATA specification statements
 G  DEFINE1 var1,var2,..., varn (optional)
 G  DEFINE2 vari, varj,..., vark (optional)
     .
     .
     .
G  DEFINE63 varx,vary,..., varz (optional)
   DATA string (optional)
G  EXECUTE
   GIRL/FORTRAN executable code (no END statement)
G  COMPLETE
```

Other GIRL/FORTRAN routines

(Purely FORTRAN routines may be included here but it is faster to add them later when the object files are linked together.)

Notes:

1) In the GIRL/FORTRAN routines, GIRL statements are declared by placing a G in Column 1. Continuation cards are handled as in FORTRAN.

2) The option card has the following entries: (the first three items must be entered in a 3I4 format)

Continuant size - Must be set to a multiple of 64, with a maximum value of 960.* This value determines the size for all continuants of all pages. It also determines the maximum number of nodes which may be defined for each page. No default.

Number of continuants to reside in the buffer - The in-core directory, the continuant size, and this item determine the size of the buffer. If the buffer, which consists of four arrays of equal size, will contain less than 64 continuants, it will have a length of:

4 * (64 + (cont. size * no. of conts. in buffer))

No default.

Highest requested page number - It is more efficient to initialize pages at the beginning of execution of a program than to create them "on demand." Value range is 1-63.

*This assumes a default prefix size of six bits and suffix size of ten bits.

The following options are in free format and must be separated by at least one blank or comma:

OUTCOR — Self explanatory. Default is the non-paged "In-Core" version of GIRS.

CREATE — Create a new graph (current default value). Note that CREATE, UPDATE, and QUERY are mutually exclusive.

UPDATE — Modify an existing graph.

QUERY — Query an existing graph.

SUFFIXnn — Allot nn bits for the identifier suffix. Default is ten bits.

$IIIIII — Declare the size of SEQ. (An integer of at most six digits preceded by a dollar sign ($).) Default size is one location.

PRINT — Print GIRL program on output file. Default is no-print.

COMMENTS — Place GIRL code with a G in Column 1 into pre-processed FORTRAN code. Default is no-comment.

LXX — Declare the maximum allowable levels of parenthesization. (An integer of at most two digits preceded by a letter L.)

NOSAVE — Eliminates the saved-index facility, and is therefore appropriate for short multivalued lists. (See the discussion of "saved index" by Zaritsky.[1])

3) Continuant specification card(s): Continuants for each page may be initialized at the beginning of execution of a program. The value range is 0-63. The format is 25I3 for all three continuant specification cards. If the "highest requested page" (see discussion on options card) has value n, then n continuant specifications are expected to be read in.

Preprocessing and Compiling a GIRL/FORTRAN Program

Assume that all the files are to reside on the system disk* and that the GIRL program "USER.GRL" is to be preprocessed and executed. The preprocessor accepts the GIRL, FORTRAN, and list file names in Command String Interpreter format with default file extension names GRL, FOR, and LST, respectively. The preprocessor will create

*The graph used by the preprocessor 'PRPGRF.BIN' must reside on the system disk drive ('SY:').

a FORTRAN file and (as an option) a GIRL listing. These files are to be named "USER.FOR" and "USER.LST," respectively. A copy of the GIRL listing will also be sent to the terminal if the PRINT option has been requested. The periods and asterisks at the beginning of lines are system prompt characters. The terminal dialog involved in preprocessing and compiling the GIRL program "USER.GRL" is as follows (linking and executing the program are described in the next section):

```
                                                     Line No.
.R PREP                                                (1)
 ALL REAL VARIABLES MUST BE DECLARED
 ERRORS ARE FLAGGED BY ****ERROR
 PLEASE ENTER FILE NAMES IN COMMAND STRING FORM
 *USER=USER                                            (2)
 or, if a list file is also desired:
 *USER,USER=USER                                       (2a)
 .R FORTRAN                                            (3)
 .USER=USER/W                                          (4)
 *^C   (control C)                                     (5)
```

DIRECT USE OF GIRS SUBROUTINES

Calls to GIRS routines may be placed directly into FORTRAN programs. Programs are compiled as with any ordinary FORTRAN program.

Linking and Executing a GIRL/FORTRAN Program

Linking a compiled GIRL program is best accomplished indirectly by executing a BATCH program which contains the link statements. If both the user's program and the GIRS routines reside on the system disk, the BATCH file "USER.BAT" appears as follows:

```
$JOB
$RUN LINK
$DATA
USER=USER,SYSLIB,RK1:FINDOP,DLETOP/F/C
NSRTOP/0:1/C
DLEXOP/0:1/C
DRCTOP/0:2/C
OPENOP/0:2/C
MSAOP/0:2/C
SETPOP/0:3/C
SAVEOP/0:3/C
SPECOP/0:3/C
PAGIOP/0:4/C
DIRIOP/0:4/C
UTILOP/0:5/C
VALUOP/0:5/C
EROP/0:5
$EOD
$EOJ
```

The following statements are needed to execute USER.BAT

```
.LOAD TT,BA
.ASS TT,LOG,LST
.R BATCH
*USER
```

The following steps should be taken if the program does not fit into main memory:

```
.R BATCH
*/U
.UNLOAD TT,BA
```

The program is now ready for execution.

```
.R USER
```

GIRS will immediately respond by printing out

```
PLEASE ENTER FILE NAMES OF OLD AND NEW GRAPHS
IN COMMAND STRING FORMAT (NEW.EXT = OLD.EXT)
.GRF IS ASSUMED EXTENSION
*NEWFIL=OLDFIL
```

Although the program may not need both "NEWFIL" and an "OLDFIL," dummy file names must be given.

OVERLAY STRUCTURE

An overlay structure has been created to reduce the effective size of out-core GIRS from 13653_{10} to 8332_{10} words. In general, the effective size cannot be reduced further due to the complex interrelationships among the subroutines as shown in Appendix B. However, under some circumstances, GIRS subroutines which perform special operations such as the creation of a new page on demand or the dumping of all GIRS system variables may be left out, reducing the size even further. These and other special operations which may be removed are discussed in note 3 in the section on limitations and memory requirements. Of course, if a user has subroutines which do not use GIRS, further space may be saved by linking them into the three overlay regions.

The sizes (in words) of the overlay regions are listed in Table 4 and the overlay structure is given in Table 5.

TABLE 4 - OVERLAY REGION SIZES

OVERLAY REGION	OCTAL	DECIMAL
Root Segment	4531	2393
1	5434	2844
2	345	229
3	4123	2131
4	412	266
5	725	469
Total	20214	8332

TABLE 5 - THE OVERLAY STRUCTURE

Overlay Region	Subroutines Listed by Segment (Size in Decimal Words)		
Root Segment	LVFDEX, LVFIND, LVFNV, LVBOTM, LVEXCH, LVDLET (2393)		
1	LVINEX LVNSRT LVUPDT (2844)	LVDLEX (481)	
2	LVMSA (229)	LVDRCT (196)	LVOPEN LVRPLC (157)
3	LVSETP LVGRN LVNPAG LVNCON (2131)	LVFECH LVDUMP LVWRIT LVCLOS (1578)	LVREOR LVOVER LVINCL (926)
4	LVPAGR LVPAGW (266)	LVVDRRD LVDRWR (99)	
5	LVALUE LVSUM (469)	LVSTAC LVPOP LVRTRN LVLFSH LVRTSH (348)	LVERR (21) [skeleton version]

LIMITATIONS AND MEMORY REQUIREMENTS

The following limitations are based on the 16-bit word size and 32K memory of a PDP-11 computer. A default suffix size of ten bits is assumed.

maximum number of pages = 63
(numbered 1 to 63)
maximum number of continuants/page = 64
(numbered 0 to 63)
prefix size = 6 bits = $2^{6} -1$ = 63
suffix size = 10 bits = $2^{10} -1$ = 1023
maximum continuant size = 960
maximum range of node values/page = 1-958

Maximum size for user program and GIRS buffer:

approximately 9900 words

Notes:

1) The GIRS buffer consists of the four arrays NODSPC, LSTSPC, LNKSPC, and FLGSPC from labeled commons LVVTR1,LVVTR2, LVVTR2, and LVVTR4, respectively.

2) The size of each array is determined as follows:

length = 64*((LVNCOR/64)+1)+LVNCOR*(LVHDRS+LVVSZE)

where LVHDRS is internally defined to two and LVVSZE (the continuant size) must be two less than a multiple of 64.

3) Special functions may be eliminated from the GIRS package if not needed. Of course, this will result in a linkage error message: UNDEF GLOBALS. The following subroutines may be considered:*

*All subroutine lengths are in decimal words.

Overlay Region 1

Segment 1

LVINEX
LVNSRT List insertion package (2844)
LVUPDT

Segment 2

LVDLEX List deletion executive (481)

Overlay Region 3

Segment 1

LVGRN Generate a random number (481)
LVNPAG Create a new page on demand (139)
LVNCON Create a new continuant on demand (227)

Segment 2

LVFECH Read-in a previously created graph (512)

LVDUMP, LVWRIT, LVCLOS Create either an ASCII dump of GIRS continuants or a binary file which contains the graph and close that channel (1066)

Segment 3

LVREOR List reorganization. Required only if lists are to be placed on specifically requested continuants (639)
LVINCL Inclusion operation (152)

Overlay Region 5

Segment 3

LVERR GIRS system variable dump (1098) Note that the LVERR routine listed in Table 5 is only a skeleton version of this routine

4) User subroutines which have no calls to GIRS routines may be added to the present overlay structure. The maximum sizes (in words) of the overlay regions are as follows:

Overlay region 1	2844
Overlay region 2	229
Overlay region 3	2131
Overlay region 4	266
Overlay region 5	469

If NODSPC, LSTSPC, LNKSPC, and FLGSPC all have lengths of 128 words and LVRWBF has a minimum length of 256 words, the minimum space required for GIRS labeled commons is 1865 words.

ADDING A USER-EMBEDDED STRATEGY

INTRODUCTION

One of the major goals of any information retrieval system is to allow efficient access to the data base, most likely by more than one user and possibly by users who do not have a sophisticated knowledge of a computing environment. The Data Base Administrator (DBA) may control this situation at the time that the information is organized and placed into the data base and also when an attempt is made to retrieve information from the data base.

If the DBA is to control the placement of information into the graph by and for several users, the DBA may wish to create a graph partition strategy which is universal to that particular set of users. As described by Berkowitz:[2]

> "A typical STRATEGY might be: "if the link is A, place the sink node on page 3; if the link is B, place the sink node on page 4; otherwise default."

An efficient graph partition would reduce disk I/O for retrievals considerably.

It is reasonable to expect users, particularly unsophisticated users, to make queries which cannot be directly answered by the data base. At the expense of some computer space and time, these queries may be handled with inferential search strategies. For example, if a particular retrieval should fail, call a retrieval strategy to determine whether the link exists at some level below the source node. This technique is described further by Zaritsky,[3] pages 46-51, and Berkowitz,[2] pages 28-44.

It is also possible for the data base to adapt to the needs of the users. For example, a monitoring strategy could be created to keep a "scorecard" of imprecise queries. Direct relationships might be placed into the graph for those queries made most often.

USE

Paged GIRS allows for the inclusion of user strategies both before and after insertion, retrieval, and deletion operations. The appropriate GIRS subroutines expect the strategies to be named as follows:

Insertion	
USRIN1	(before)
USRIN2	(after)
Retrieval	
USRFD1	(before)
USRFD2	(after)
Deletion	
USRDL1	(before)
USRDL2	(after)

The following switches (all from labeled common LVSWIT) are needed to use the strategy. Variables with "IN" in the name are used for insertion, those with "FD" for retrieval, and those with "DL" for deletion.

The "before" strategies may be skipped if LVIN1, LVFD1, or LVDL1 are .FALSE. (default). The "after" strategies may be skipped if LVIN2, LVFD2, or LVDL2 are .FALSE. (default). The insertion, retrieval, or deletion operations may be skipped entirely if LVIN4, LVFD4, or LVDL4 are set in USRIN2, USRFD1, or USRDL1, respectively, to .FALSE. (default is .TRUE.). If the "before" strategies are called, LVIN2, LVFD2, or LVDL2, may be modified by LVIN3, LVFD3, or LVDL3, respectively. The user strategies may not be used recursively.

PROPOSED EXTENSIONS

In the near future, we hope to merge the paged version of GIRS with a paged hardware associative memory facility. The result will be an enhanced system with high speed relational processing.

ACKNOWLEDGMENTS

The overall scheme for the software described was designed by Dr. S. Berkowitz. It was written under his supervision with his advice and encouragement.

APPENDIX A

VARIABLES IN LABELED COMMON

In the following list of all the labeled commons required by out-core GIRS, the external names, as created by the GIRL preprocessor, are given for those commons which must be included in the user's main program. Otherwise, internal names are used.

```
EXTERNAL:
COMMON /LVARGS/ LVFUNC,LVVARG,LVVPOS,LVVTYP,LVVAL,LVVNVL,LVSKIP,
1               LVVTR,LVVINC,LVNDXN,LVVALS(10),LVTYPE(10),
2               LVSRSF,LVLNSF,LVSNSF,LVNTYP
COMMON /LVVSEQ/ LVSIZE,LVSEQ1,LVSEQ2,SEQSPC(1)
COMMON /LVRAND/ LVKPRM,LVKS,LVKY,LVKDY,LVKDX,LVTEMP,LVLIST,
1               LVNTBL(256)
COMMON /LVVTR1/ NODSPC(buffer size)
1      /LVVTR2/ LSTSPC(buffer size)
2      /LVVTR3/ LNKSPC(buffer size)
3      /LVVTR4/ FLGSPC(buffer size)
COMMON /LVCRNT/ LVVGSP,LVCTRL,LVCTR1,LVLSTV,LVNFRE,LVFREE,LVDREG,
2               LVVMSA,LVPGLC,LVCRNT
COMMON /LVBUFR/ LVVSZE,LVNWCH,LVOLCH,LVCMPR,LVPGHD,LVBFSZ,
1               LVDRSZ,LVNCOR,LVHDRS,LVMSAD,
2               LVSFSZ,LVBKSZ,LVDRBK,LVPGH4
COMMON /LVREGS/ LVCUPG(4),LVREQP(4),LVLVPG(4),LVMSAR
1               LVHRPG,LVNMSA,LVHAPG(2),LVRCNT,LVUCNT,LVDRPG,
2               LVDIRC,LVOTLC,LVOTDR(256),
3               LVRWBF(4*continuant size)
COMMON /LVPRAM/ LVBFLC,LVLNTH,LVVERR,LVERNO,LVBNRY,LVBCD,
1               LVMODE,LVPRS,LVLUN
COMMON /LVRUN/  LVRNTP,LVCORE
COMMON /LVSTAK/ LVLEVL,LVNVAR,LVSTAK(140)
COMMON /LVMASK/ LVWRIT,LVNUSE,LVNWCN,LVMSK3,LVMSSF,LVMSPF
COMMON /LVSWIT/ LVSTP,LVSNGL,LVNXTR,LVIN1,LVIN2,LVFD1,
1               LVFD2,LVDL1,LVDL2,LVIN3,LVFD3,LVDL3,LVDMP,
2               LVFD4,LVDL4,LVIN4
COMMON /LVUSER/ USER(228)
COMMON /LVUTIL/ FILSPC(39),DEFEXT(2)
```

INTERNAL:

```
COMMON /LVFLAG/ FLOMSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,
1               FLG67,FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,
2               FLAG14,FLAG15
COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
1               USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
COMMON /LVFND/  IADD,THIS,LSTHED,LOC,LAST,LASTLC
COMMON /LVFND/  COUNT,ABSPOS,LSTCON
COMMON /LVINS1/ REORG,FULL,RPLACE
COMMON /LVDEL1/ NUMRET
```

Note that all variables from labeled common LVSWIT must be set to LOGICAL*1.

APPENDIX B

SUBROUTINE CALLING STRUCTURE

This appendix lists all the subroutines in Out-Core GIRS and the GIRS subroutines called by them:

```
SUBROUTINE LVSETP
        LVFECH
        LVGRN()
        LVPAGW
        LVDRWR
        LVMSA()
        LVPAGR()
        LVFIND
        LVNSRT

SUBROUTINE LVMSA(CONNUM)
        LVERR()
        LVDRRD()

SUBROUTINE LVCLOS
        LVERR()
        LVPAGW
        LVDRWR

SUBROUTINE LVDRRD(CHAN)
        LVERR()

SUBROUTINE LVDRWR
        LVERR()

SUBROUTINE LVPAGR(CHAN)
        LVERR()

SUBROUTINE LVGRN(NODE)
        LVLFSH(,)
        LVEXCH
        LVERR()

SUBROUTINE LVEXCH
        LVDRCT
        LVMSA()
        LVOPEN
        LVPAGR()
        LVRPLC
        LVSUM

SUBROUTINE LVSTAC
        LVERR()
```

```
SUBROUTINE LVPOP
        LVERR()

SUBROUTINE LVDRCT
        LVSTAC
        LVFIND
        LVPOP

FUNCTION LVLFSH(WORD,BITS)

FUNCTION LVRTSH(WORD,BITS)

SUBROUTINE LVDLEX
        LVSTAC
        LVPOP
        LVFDEX(,,,,)
        LVDLET
        LVERR()

SUBROUTINE LVDLET

SUBROUTINE LVRTRN

SUBROUTINE LVFDEX(INDEX,INDXAD,KFUNC,KARG,SAVCON)
        LVSTAC
        LVPOP
        LVRTSH(,)
        LVEXCH
        LVFIND
        LVFNV(,,,,)
        LVEXCH
        LVBOTM

SUBROUTINE LVFIND
        LVERR()

SUBROUTINE LVFNV(INDEX,INDXAD,KFUNC,KARG,SAVCON)

SUBROUTINE LVBOTM
        LVEXCH
        LVERR()
        LVFIND

SUBROUTINE LVINCL
        LVFDEX(,,,,)
```

```
SUBROUTINE LVINEX
        LVSTAC
        LVPOP
        LVNPAG
        LVGRN()
        LVLFSH(,)
        LVRTSH(,)
        LVERR()
        LVEXCH
        LVNCON
        LVFDEX
        LVREOR()
        LVNSRT
        LVFIND
        LVOVER

SUBROUTINE LVNSRT
        LVUPDT
        LVFIND
        LVFNV(,,,,)

SUBROUTINE LVUPDT

SUBROUTINE LVREOR(REQCON)
        LVERR()
        LVEXCH
        LVSTAC
        LVFIND
        LVNSRT
        LVDLET
        LVPOP

SUBROUTINE LVOVER
        LVSTAC
        LVDLET
        LVPOP
        LVNSRT

SUBROUTINE LVNPAG
        LVMSA()
        LVEXCH
        LVOPEN
        LVSETP
        LVPAGW
        LVRPLC
        LVSUM
```

```
SUBROUTINE LVNCON
        LVMSA()
        LVOPEN
        LVSETP
        LVPAGW
        LVPAGR()
        LVRPLC
        LVSUM

SUBROUTINE LVERR(DUMP)

SUBROUTINE LVOPEN
        LVALUE
        LVPAGW

SUBROUTINE LVRPLC
        LVSTAC
        LVFIND
        LVDLET
        LVNSRT
        LVPOP

SUBROUTINE LVSUM

SUBROUTINE LVALUE
        LVDUMP()

SUBROUTINE LVFECH
        LVERR()
        LVPAGR()
        LVDRRD()
        LVDRWR
        LVMSA()
        LVPAGW

SUBROUTINE LVDUMP(DUMP)
        LVERR()
        LVWRIT(,)
        LVEXCH
        LVPAGW
        LVCLOS

SUBROUTINE LVWRIT(NBIAS,NUMBLK)
```

APPENDIX C

SUBROUTINE LISTINGS

```
C
C
C
      SUBROUTINE LVFDEX(INDEX,INDXAD,KFUNC,KARG,SAVCON)
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1       DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,FD1TMP,
     2       DL2TMP,IN2TMP,FD2TMP,REORG,FULL,LSTCON,RPLACE
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1             INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2             LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1             FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2             FLAG15
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1             MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1             INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1             USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1             DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2             FINDFL,DLETFL,NSRTFL
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1             LUN
      COMMON /LVFND/       COUNT,ABSPOS,LSTCON
      COMMON /LVINS1/ REORG,FULL,RPLACE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
D     PAUSE 'IN LVFDEX'
C THE PURPOSE OF THE FIND EXECUTIVE ROUTINE IS TO BRING THE PROPER
C CONTINUANT INTO THE BUFFER.  IF THE PROGRAMMER DOES NOT SPECIFY THE
C CONTINUANT, ALL OF THE CONTINUANTS OF THAT PAGE WILL BE SEARCHED UNTIL
C EITHER A VALUE IS FOUND OR ALL OF THE CONTINUANTS HAVE BEEN LOOKED AT.
C     IF THE CONTINUANT HAS BEEN SPECIFIED, THE SEARCH WILL PROCEED
C TO THE NEXT CONTINUANT ONLY IF FLAG 11 HAS BEEN SET FOR THAT LIST.
C     IF THE CONTINUANT HAS BEEN SPECIFIED, THE SEARCH WILL PROCEED
C TO THE PREVIOUS CONTINUANT ONLY IF FLAG 10 HAS BEEN SET FOR THAT LIST.
C     IF FD1STR IS .TRUE. USER STRATEGY ROUTINE USRFD1 PRECEEDS
C RETRIEVAL ACTION AND CONTINUES IF FINDFL IS .TRUE.
C USRFD2 IS CALLED AFTER THE RETRIEVAL IF FD2STR IS SET TO .TRUE.
C
      XXX=1000
      IF((FLGSPC(CTRL1+REGASP).OR.FL3MSK).NE.FL3MSK)XXX=XXX*XXX
      REG=NODSPC(DIRSZE+REGAS)
      USE=LNKSPC(DIRSZE+USECT)
C     IF(USE.GT.440) XXX=XXX*XXX
C     IF((FLGSPC(67).EQ.0).AND.(FLGSPC(144).EQ.0)) XXX=XXX*XXX
      COUNT = 0
      ABSPOS = IABS(IPOS)
      C
      C CALL USER'S FIRST RETRIEVAL STRATEGY ROUTINE ?
0023        IF(FD1STR .EQ. .FALSE.) GO TO 100
      C
      C TO PREVENT RECURSION, INHIBIT FURTHER CALLS TO USER STRATEGY ROUTINES
0025        FD1TMP = FD1STR
0026        FD2TMP = FD2STR
0027        FD1STR = .FALSE.
0028        FD2STR = .FALSE.
0029        DL1TMP = DL1STR
0030        DL2TMP = DL2STR
0031        DL1STR = .FALSE.
0032        DL2STR = .FALSE.
0033        IN1TMP = IN1STR
0034        IN2TMP = IN2STR
0035        IN1STR = .FALSE.
0036        IN2STR = .FALSE.
      C
      C SET UP FOR FIRST USER ROUTINE
0037        CALL LVSTAC
0038        CALL USRFD1
0039        CALL LVPOP
0040        FD1STR = FD1TMP
0041        FD2STR = FD2TMP
0042        DL1STR = DL1TMP
0043        DL2STR = DL2TMP
0044        IN1STR = IN1TMP
0045        IN2STR = IN2TMP
      C
      C PROCEED WITH RETRIEVAL ?
0046        IF(FINDFL .EQ. .FALSE.) GO TO 600
      C
      C REQPAG(2) IS SET IN CALLING PROGRAM.  DEFAULT IS -2 ("ANY" CONTINUANT)
      C SEPARATE PREFIX AND SUFFIX FROM SOURCE NODE (IARG) AND LINK (IFUNC)
0048  100   IF(IARG .LT. 2**SUFSZE) RETURN
0050        REQPAG(1) = LVRTSH(IARG .AND. MASKPF, SUFSZE)
0051        SRCSUF          = IARG .AND. MASKSF
0052        IF(IFUNC .LT. 2**SUFSZE) RETURN
0054        REQPAG(3) = LVRTSH(IFUNC .AND. MASKPF, SUFSZE)
0055        LNKSUF          = IFUNC .AND. MASKSF
      C
      C IS SAVED INDEX OPTION ON ?
0056        IF(NSKIP .EQ. 1) GO TO 150
0058        REQPAG(2) = SAVCON
      C
      C REQPAG(2) IS SET AT THE END OF LVFIND TO -2, IF IT IS NOT RESET BY THE
      C PROGRAMMER FOR A RETRIEVAL, THEN THE REQUESTED CONTINUANT IS SET TO
      C ZERO AND A SEARCH OF ALL CONTINUANTS IS ALLOWED.
      C
0059  150   REQCON = REQPAG(2)
0060        IF(REQPAG(2) .EQ. -2) REQPAG(2) = 0
      C
      C*** BRING THE REQUESTED PAGE, CONTINUANT INTO CORE.
      C    MAKE IT THE CURRENT PAGE, CONTINUANT.
      C
0062        ITESTR = -1
0063  200   CALL LVEXCH
```

```
C
C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED
      IF(MSARET .LE. 0) GO TO 600
C
C DESIRED PAGE, CONTINUANT IS NOW IN PLACE.
C ASSUME LIST DOES NOT CONTINUE BEYOND PRESENT CONTINUANT
      LSTCON = .FALSE.
C
C SEARCH FOR FUNCTION HEAD.
      CALL LVFIND
C
C FLAG CONTINUANT AS USED
      FLGSPC(CTRLPT+HDRFLG) = FLGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSD
      LNKSPC(CTRLPT+USECT)  = LNKSPC(CTRLPT+USECT) + 1
C
C HAS THE FUNCTION HEAD BEEN FOUND ?
      IF(ITESTR .GT. 0) GO TO 300
C
C IF THE CONTINUANT IS NOT SPECIFIED, EXAMINE NEXT CONTINUANT.
      IF(REQCON .NE. -2) GO TO 600
      REQPAG(2) = REQPAG(2) + 1
      GO TO 200
C
C FUNCTION HEAD FOUND
C SEARCH FROM TOP OR BOTTOM OF LIST ?
300   IF(IPOS) 500,600,410
C
400   CALL LVFIND
C
C DOES A PORTION OF THE CORRECT LIST RESIDE ON THIS CONTINUANT ?
      IF(ITESTR .LT. 0) GO TO 450
C
C BEGIN SEARCH DOWN THE LIST
410   CALL LVFNV(INDEX,INDXAD,KFUNC,KARG,SAVCON)
C
C FLAG CONTINUANT AS USED
      FLGSPC(CTRLPT+HDRFLG) = FLGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSD
      LNKSPC(CTRLPT+USECT)  = LNKSPC(CTRLPT+USECT) + 1
C
C SUCCESSFUL RETRIEVAL ?
      IF(ITESTR .GT. 0) GO TO 600
C
C DOES THE LIST EXTEND TO ANOTHER CONTINUANT ?
      IF(LSTCON .EQ. .FALSE.) GO TO 600
C
C UPDATE REQUESTED CONTINUANT AND BRING INTO THE BUFFER
450   REQPAG(2) = REQPAG(2) + 1
      CALL LVEXCH
C
C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED ?
      IF(MSARET .LE. 0) GO TO 600
      GO TO 400
C
C SEARCH FROM THE BOTTOM OF THE LIST
C   BRING IN CONTINUANT CONTAINING LAST PORTION OF MVI
500   CALL LVBOTM
      GO TO 530
```

```
C
520   CALL LVFIND
C
C DOES A PORTION OF THE CORRECT LIST RESIDE ON THIS CONTINUANT ?
      IF(ITESTR .LT. 0) GO TO 550
C
C BEGIN SEARCH UP THE LIST
530   CALL LVFNV(INDEX,INDXAD,KFUNC,KARG,SAVCON)
C
C FLAG CONTINUANT AS USED
      FLGSPC(CTRLPT+HDRFLG) = FLGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSD
      LNKSPC(CTRLPT+USECT)  = LNKSPC(CTRLPT+USECT) + 1
C
C SUCCESSFUL RETRIEVAL ?
      IF(ITESTR .GT. 0) GO TO 600
C
C DOES THE LIST EXTEND TO ANOTHER CONTINUANT ?
      IF(LSTCON .EQ. .FALSE.) GO TO 600
C
C UPDATE REQUESTED CONTINUANT
550   REQPAG(2) = REQPAG(2) - 1
C
C HAVE ALL CONTINUANTS OF REQ(PAGE) BEEN EXAMINED ?
      IF(REQPAG(2) .LT. 0) GO TO 600
C
C BRING REQ(P,C) INTO THE BUFFER
      CALL LVEXCH
      GO TO 520
C
C CALL SECOND USER RETRIEVAL STRATEGY ROUTINE ?
600   IF(FD2STR .EQ. .FALSE.) GO TO 700
      FD1TMP = FD1STR
      FD2TMP = FD2STR
      FD1STR = .FALSE.
      FD2STR = .FALSE.
      DL1TMP = DL1STR
      DL2TMP = DL2STR
      DL1STR = .FALSE.
      DL2STR = .FALSE.
      IN1TMP = IN1STR
      IN2TMP = IN2STR
      IN1STR = .FALSE.
      IN2STR = .FALSE.
      CALL LVSTAC
      CALL USRFD2
      CALL LVPOP
      FD1STR = FD1TMP
      FD2STR = FD2TMP
      DL1STR = DL1TMP
      DL2STR = DL2TMP
      IN1STR = IN1TMP
      IN2STR = IN2TMP
C
C RESET 'REQUESTED CONTINUANT' DEFAULT TO 'ANY'
700   REQPAG(2) = -2
C RESET TO DEFAULT VALUES
      IPOS = 1
      ITYP = 3
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVFIND
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1      DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,
     2      DL2TMP,IN2TMP,FD2TMP,REORG,FULL,LSTCON,RPLACE
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1          INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2          LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1          HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2          DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1          FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2          FLAG15
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1          MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1          INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1          USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1          DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2          FINDFL,DLETFL,NSRTFL
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1          LUN
      COMMON /LVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /LVFND/       COUNT,ABSPOS,LSTCON
      COMMON /LVINS1/ REORG,FULL,RPLACE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
C IADD   = (RELATIVE) COMPUTED FUNCTION ADDRESS
C THIS   = (RELATIVE) LOCATION OF FUNCTION ON CONFLICT LIST
C LOC = (RELATIVE) LOCATION OF RETRIEVED VALUE
C LSTHED = -1, SINGLE VALUED LIST
C      =  0, NO LIST IS FOUND
C      >  0, (RELATIVE) ADDRESS OF FIRST VALUE
C ITESTR =  1, RETRIEVAL IS SUCCESSFUL (IVAL = RETURNED VALUE)
C      = -1, RETRIEVAL IS FAILURE (IVAL = SOURCE NODE)
C
D     PAUSE 'IN LVFIND'
      ITESTR = 1
      IADD = SRCSUF + LNKSUF
      IF(IADD .GT. PAGSZE) IADD = IADD-PAGSZE
      IF(IADD .LE. PAGSZE) GO TO 2
C
C IFUNC OR IARG ARE INCORRECT, STOP
      TYPE 3, IFUNC,IARG
3     FORMAT(//,' ****ERROR****  LINK ',I5,'  OR SOURCE NODE ',I5,' ARE
     1 UNDEFINED',/)
      ERRNUM = 40
      DUMP = 0
      CALL LVERR(DUMP)
      STOP
C
2     LSTHED = 0
      THIS = IADD
      IF((FLGSPC(CTRL1 + THIS) .AND. FL5MSK) .EQ. 0) GO TO 99
C
C SEARCH CONFLICT LIST FOR KEY (IFUNC OR LINK)
1     IF(NODSPC(CTRL1 + THIS) .EQ. IFUNC) GO TO 4
      LAST = THIS
      THIS = LNKSPC(CTRL1 + THIS)
      IF((FLGSPC(CTRL1 + THIS) .AND. FL5MSK) .NE. 0) GO TO 99
      GO TO 1
C
C THE FUNCTION HAS BEEN FOUND.
C TEST FOR SINGLE VALUE LIST (SVL)  OR MULTIVALUED LIST (MVL).
4     IF((FLGSPC(CTRL1 + THIS) .AND. FL0MSK) .NE. 0)  GO TO 14
C
C SINGLE VALUED LIST.
      LSTHED = -1
      LOC = THIS
      IVAL = LSTSPC(CTRL1 + LOC)
      RETURN
C
C MULTIVALUED LIST.  OBTAIN FIRST VALUE.
14    LSTHED = LSTSPC(CTRL1 + THIS)
      LOC = LSTHED
      IVAL = NODSPC(CTRL1 + LOC)
      LASTLC = LNKSPC(CTRL1 + LSTHED)
      RETURN
C
C FUNCTION IS NOT ON THIS CONTINUANT
99    ITESTR = -1
      IVAL = IARG
      RETURN
      END
      C
      C
      C
0001        SUBROUTINE LVFNV(INDEX,INDXAD,KFUNC,KARG.SAVCON)
0002        IMPLICIT INTEGER(A-Z)
0003        LOGICAL*1 SNGI BK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR.DL1STR.
           1      DL2STR,DUMPFI ,CURENT,FINDFI ,DLETFI ,NSRTFI ,
           2      DL2TMP,IN2TMP,FD2TMP,REORG,FULL,LSTCON,RPLACE
0004        COMMON /I VARGS/ IFUNC,IARG.IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR.
           1              INCI UD,INDXON.IVALS(10),ITYP1(10),SRCSUF,
           2              LNKSUF,SNKSUF.INSTYP
0005        COMMON /I VREGS/ CURPAG(4),RFQPAG(4),LSTVPG(4),MSARET,
           1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
           2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
0006        COMMON /I VMASK/ MWRITE,NOTUSD,NEWCON.FI GMSK,MASKSF,MASKPF
0007        COMMON /I VFI AG/ FI 0MSK,FI 1MSK,FI 2MSK,FI 3MSK,FI 4MSK,FI 5MSK,FI G67,
           1              FI AG8,FI AG9,FI AG10,FI AG11,FI AG12.FI AG13,FI AG14,
           2              FI AG15
0008        COMMON /I VCRNT/ REGASP,CTRI PT,CTRI 1,LEASTV,NTFREE,FREE,DREGSP,
           1              MSA,PAGI OC,CURENT
0009        COMMON /I VBUFR/ PAGSZE,NWCHAN,OI CHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
           1              INCORE,HDRSZE,MSADIR.SUFSZE,BI KSZE,DIRBI K,PAGHD4
0010        COMMON /I VHDVI/ THSMSA,REGAS,PAGENO,CONTNO.INSDEL,
           1              USECT,HDRFI G,READVI ,OI DNDH,DNODEH,NROWH,DROWH
0011        COMMON /I VSWIT/ SETUP,SNGI BK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR.
           1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFI ,
           2              FINDFI ,DLETFI ,NSRTFI
0012        COMMON /I VPRAM/ BUFI OC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES.
           1              LUN
0013        COMMON /I VADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
0014        COMMON /I VFND/         COUNT,ABSPOS,LSTCON
0015        COMMON /I VINS1/ REORG,FULL,RPLACE
0016        COMMON /I VVTR1/ NODSPC(1)
           1    /I VVTR2/ LSTSPC(1)
           2    /I VVTR3/ LNKSPC(1)
           3    /I VVTR4/ FI GSPC(1)
0017        DATA NFI AG4/'177767/
      C
      C LVFIND MUST BE CALLED IMMEDIATELY PRIOR TO THE CALL TO THIS ROUTINE
      C INPUT IS EXPECTED THRU COMMONS LVARGS, LVFND, AND LVADDR.  THIS ROUTINE
      C SEARCHES THE MULTIVALUE LIST FOR THE IPOS'TH VALUE OF THE REQUESTED
      C TYPE.  IF SVI , TYPE MUST BE EITHER UNSPECIFIED OR CORRFCT.
      C
      C DOES THE FUNCTION EXIST ?
      D      PAUSE 'IN LVFNV'
0018         IF(ITESTR .IT. 0) GO TO 700
0020         IF(LSTHED .GT. 0) GO TO 100
      C SVI  -  DOES FUNCTION QUALIFY ?
0022         IF(ABSPOS .NE. 1) GO TO 699
0024         IF(ITYP .EQ. 3) GO TO 700
0026         ISTYP = (FI GSPC(CTRI 1 + LOC) .AND. FI G67)
0027         IF(ISTYP .EQ. 3) ISTYP = 2
0029         IF(ISTYP .NE. ITYP)  GO TO 699
0031         GO TO 700
      C
      C MVI - FIRST VALUE HAS ALREADY BEEN FOUND BY LVFIND
0032  100    IF(IPOS .EQ. 1 .AND. ITYP .EQ. 3) GO TO 500
```

```
      C
      C *** BEGIN SEARCH
      C IF THE SAVED INDEX FACILITY IS NOT TO BE USED. GO TO 200
0034  120   IF(NSKIP .EQ. 1) GO TO 200
0036        IF(INDEX .EQ. 0) GO TO 200
      C
      C SAVED INDEX CAN'T BE USED IF IMMEDIATE PAST HISTORY = .  EXED
      C INSERTION OR DELETION.
0038        IF((FLGSPC(CTRL1 + THIS) .AND. FL4MSK)  .NE.  0) GO TO 200
      C
      C SAVED INDEX CAN'T BE USED IF SOURCE NODE OR LINK HAVE BEEN CHANGED
0040        IF((KFUNC .NE. IFUNC) .OR. (KARG .NE. IARG)) GO TO 200
      C
      C SAVED INDEX CAN'T BE USED IF DIRECTION OF SEARCH HAS SWITCHED
0042        IF((IPOS*INDEX) .LE. 0) GO TO 200
0044        NDX = FLGSPC(CTRL1 + INDXAD)
      C
      C SAVED INDEX CAN'T BE USED IF VALUE AT SAVED INDEX HAS BEEN MOVED
0045        IF((NDX .AND. FL5MSK) .NE. 0)  GO TO 200
      C
      C SAVED INDEX CAN'T BE USED IF VALUE AT SAVED INDEX HAS BEEN REMOVED
0047        IF((NDX .AND. FL1MSK) .EQ. 0)  GO TO 200
      C
      C IS SEARCH FROM BEGINNING FASTER THAN FROM SAVED INDEX ?
0049        KNDEX = IABS(INDEX)
0050        IF(ABSPOS .LT. 2) GO TO 200
0052        IF((ABSPOS+ABSPOS) .LE. KNDEX)  GO TO 200
      C
      C SAVED INDEX CAN BE USED, BEGIN SEARCH AT INDXAD.
0054        LOC = INDXAD
      C FIND RELATIVE DISTANCE FROM SAVED INDEX AND DETERMINE WHETHER TO
      C COUNT UP OR DOWN.  IF REQUESTED POSITION IS CLOSER TO THE BEGINNING
      C OF THE LIST THAN THE SAVED INDEX, COUNT UP, OTHERWISE, COUNT DOWN.
      C
0055        LENGTH = INDEX-IPOS
0056        ABSPOS = IABS(LENGTH)
0057        IF(LENGTH) 300,450,170
      C
      COUNT UP FROM INDXADD
      C
0058  170   ITOP = 0
0059        GO TO 420
      C
      C DO NOT USE SAVED INDEX.  START FROM THE BEGINNING OR END OF LIST
      C
0060  200   FLGSPC(CTRL1 + THIS) = FLGSPC(CTRL1 + THIS) .AND. NFLAG4
0061        IF(IPOS) 400,699,320
      C
      COUNT DOWN
      C
0062  300   LASTLC = LOC
0063        LOC = LSTSPC(CTRL1 + LOC)
0064        IF((FLGSPC(CTRL1 + LOC) .AND. FL0MSK) .NE. 0)  GO TO 600
0066  320   IF(ITYP .EQ. 3) GO TO 330
0068        ISTYP = (FLGSPC(CTRL1 + LOC) .AND. FLG67)
0069        IF(ISTYP .EQ. 3)ISTYP = 2
0071        IF(ISTYP .NE. ITYP) GO TO 300
```

```
330    COUNT = COUNT+1
       IF(COUNT .NE. ABSPOS) GO TO 300
       GO TO 450
C
COUNT UP FROM THE BOTTOM OF THE LIST
C
400    ITOP = 1
420    LOC = LNKSPC(CTRL1 + LOC)
       IF(ITOP .EQ. 1) GO TO 430
       IF((FLGSPC(CTRL1 + LSTSPC(CTRL1 + LOC)) .AND. FLGMSK) .NE. 0)
      1 GO TO 650
430    ITOP = 0
       IF(ITYP .EQ. 3) GO TO 440
       ISTYP = (FLGSPC(CTRL1 + LOC) .AND. FLG67)
       IF(ISTYP .EQ. 3)ISTYP = 2
       IF(ISTYP .NE. ITYP) GO TO 420
440    COUNT = COUNT+1
       IF(COUNT .NE. ABSPOS) GO TO 420
450    IVAL = NODSPC(CTRL1 + LOC)
C
C  SAVE INDEX PARAMETERS AFTER SUCCESSFUL RETRIEVAL
C
500    IF(NSKIP .EQ. 1) GO TO 700
       KARG   = IARG
       KFUNC  = IFUNC
       INDXAD = LOC
       INDEX = IPOS
       SAVCON = CURPAG(2)
       GO TO 700
C
C POSSIBLE FAILURE.  DOES MVL EXTEND FORWARD TO ANOTHER CONTINUANT
600    IF((FLGSPC(CTRL1 + LASTLC) .AND. FLAG11) .EQ. 0) GO TO 699
       LSTCON = .TRUE.
       GO TO 699
C
C POSSIBLE FAILURE.  DOES MVL EXTEND BACKWARD TO ANOTHER CONTINUANT
650    IF((FLGSPC(THIS) .AND. FLAG10) .EQ. 0) GO TO 699
       LSTCON = .TRUE.
C
C FAILURE EXIT
699    ITESTR = -1
       IF(NSKIP .EQ. 0) INDEX = 0
       IVAL = IARG
C
C SUCCESS EXIT, SET DEFAULTS.
700    ITYP = 3
       RETURN
       END
```

```
C
C
C
      SUBROUTINE LVBOTM
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1       DL2STR,DUMPFI,CURENT,FINDFI,DLETFI,NSRTFI,BAKCON,
     2       DL2TMP,IN2TMP,FD2TMP,REORG,FULL,LSTCON,RPLACE
      COMMON /IVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1             INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2             LNKSUF,SNKSUF,INSTYP
      COMMON /IVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /IVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /IVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1             FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2             FLAG15
      COMMON /IVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1             MSA,PAGLOC,CURENT
      COMMON /IVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1             INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /IVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1             USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /IVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1             DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFI,
     2             FINDFI,DLETFI,NSRTFI
      COMMON /IVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1             LUN
      COMMON /IVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /IVFND/        COUNT,ABSPOS,LSTCON
      COMMON /IVINS1/ REORG,FULL,RPLACE
      COMMON /IVDEL1/ NUMRET,BAKCON
      COMMON /IVVTR1/ NODSPC(1)
     1    /IVVTR2/ LSTSPC(1)
     2    /IVVTR3/ LNKSPC(1)
     3    /IVVTR4/ FLGSPC(1)
C
C THIS ROUTINE BRINGS INTO THE BUFFER THE LAST CONTINUANT OF A PAGE WHICH
C CONTAINS A PORTION OF THE REQUESTED LIST.
C ASSUME THAT THE BUFFER CONTAINS THE CONTINUANT WHICH HOLDS THE FIRST
C PORTION OF THE MVL.
C
D     PAUSE 'IN LVBOTM'
      TMPREQ = CURPAG(2)
C SVI ?
100   IF(LSTHED .GT. 0) GO TO 120
      LASTLC = THIS
      GO TO 140
C
C GET FIRST "VALUE" ON MVL
120   ISTLOC = LSTSPC(CTRL1 + THIS)
C
C GET LAST "VALUE" ON MVL
      LASTLC = LNKSPC(CTRL1 + ISTLOC)
C
C DOES THE LIST END ON THIS CONTINUANT ?
```

```
140   IF((FLGSPC(CTR11 + LASTLC) .AND. FLAG11) .EQ. 0) RETURN
      LSTCON = .TRUE.
C
C EXAMINE NEXT (SEQUENTIAL) CONTINUANT FOR A PORTION OF THE MVI
200   REQPAG(2) = REQPAG(2) + 1
      CALL LVEXCH
C
C ERROR IF SET OF CONTINUANTS IS EXHAUSTED
      IF(MSARET .GT. 0) GO TO 250
C NO ERROR IF SEARCH ORIGINATED FROM LVULEX
      IF(BAKCON .EQ. .FALSE.) GO TO 220
      REQPAG(2) = TMPREQ
      RETURN
C
220   ERRNUM = 42
      DUMP = 0
      CALL LVERR(DUMP)
      STOP
C
C DOES THIS CONTINUANT CONTAIN A PORTION OF THE MVI ?
250   CALL LVFIND
      IF(ITESTR .LT. 0) GO TO 200
      TMPREQ = REQPAG(2)
      GO TO 100
      END
```

```
C
C
C
      SUBROUTINE LVINEX
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1      DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,FD1TMP,
     2      DL2TMP,IN2TMP,FD2TMP,INSIDE,FULL,REORG,LSTCON,NXTCON,
     3      RPLACE
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR.
     1            INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2            LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1            HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2            DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK.FL3MSK.FL4MSK,FL5MSK,FLG67,
     1            FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2            FLAG15
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1            MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1            INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1            USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR.FD1STR,FD2STR.
     1            DL1STR.DL2STR.IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2            FINDFL,DLETFL,NSRTFL
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1            LUN
      COMMON /LVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /LVFND/        COUNT,ABSPOS,LSTCON
      COMMON /LVINS1/ REORG.FULL,RPLACE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
      DATA INSIDE /.FALSE./
C
C THE INSERT EXECUTIVE ROUTINE COMPLETES THE TRIPLE IF NECESSARY AND
C OBTAINS THE CORRECT P,C FOR SUBROUTINE LVNSRT TO OPERATE ON.
C
C IS LVNSRT BEING CALLED FROM AN INSERT STRATEGY ROUTINE ?
D     PAUSE 'IN LVINEX'
      IF(INSIDE .EQ. .TRUE.) GO TO 100
C
C TO PREVENT RECURSION. SAVE THE FIND STRATEGY FLAGS AND TURN THEM OFF
      FD1TMP = FD1STR
      FD2TMP = FD2STR
      FD1STR = .FALSE.
      FD2STR = .FALSE.
C
C CALL USER'S FIRST INSERT STRATEGY ROUTINE ?
      IF(IN1STR .EQ. .FALSE.) GO TO 100
C
C TO PREVENT RECURSION, INHIBIT CALLS TO ALL USER STRATEGY ROUTINES
```

```
      IN1TMP = IN1STR
      IN2TMP = IN2STR
      IN1STR = .FALSE.
      IN2STR = .FALSE.
      DL1TMP = DL1STR
      DL2TMP = DL2STR
      DL1STR = .FALSE.
      DL2STR = .FALSE.
C
C SET UP FOR FIRST USER ROUTINE
      CALL LVSTAC
      INSIDE = .TRUE.
      CALL USRIN1
      INSIDE = .FALSE.
      CALL LVPOP
      IN1STR = IN1TMP
      IN2STR = IN2TMP
      DL1STR = DL1TMP
      DL2STR = DL2TMP
C
C PROCEED WITH INSERTION ?
      IF(NSRTFL .EQ. .FALSE.) GO TO 1000
C
C *** ENSURE THAT THE TRIPLE IS COMPLETELY DEFINED ***
C *** BRING IN REQ(P,C), DEFINE AS CURRENT(P,C)
C     TEST SOURCE NODE
C     IARG = -1       PLACE ON NEW PAGE AND DEFINE
C        =       0            PLACE ON CURRENT PAGE AND DEFINE
C       >=  2**SUFSZE, ALREADY DEFINED, SEPARATE PREFIX AND SUFFIX
C        =N<2**SUFSZE, PLACE ON PAGE N AND DEFINE
100   NXTCON = .FALSE.
      REQCON = REQPAG(2)
      IF(IARG .EQ. -1) REQCON = -1
      IF(IARG) 110,120,130
C
C PLACE ON NEW PAGE (CONT = 0)
110   CALL LVNPAG
C
C PLACE ON CURRENT PAGE (AND CONT) AND DEFINE SUFFIX
120   CALL LVGRN(SRCSUF)
C
C RECONSTRUCT IARG
      IARG = LVIFSH(CURPAG(1),SUFSZE) .OR. SRCSUF
      GO TO 200
C
C A SPECIFIC PAGE IS REQUESTED
C     IS ONLY THE SUFFIX DEFINED ?  (IF SO, IT IS A PAGE REQUEST W/O SUF)
130   IF(IARG .GE. 2**SUFSZE) GO TO 140
      REQPAG(1) = IARG
      GO TO 145
140   REQPAG(1) = LVRTSH(IARG.SUFSZE)
C
C IMPROPER PAGE REQUEST ?
      IF(REQPAG(1) .LE. HACTPG(1)) GO TO 145
      ERRNUM = 60
      DUMP = 0
      CALL LVERR(DUMP)
```

```
0064          STOP
      C
      C TEST "REQUESTED CONTINUANT"
      C   REQPAG(2) >= 0, CONTINUANT SPECIFIED
      C     -1, NEW CONTINUANT
      C     -2, "ANY" CONTINUANT
      C     -3, CURRENT CONTINUANT IF CURRENT PAGE = REQUESTED PAGE
0065  145     IF(REQPAG(2) .GE. 0) GO TO 155
0067          IF(REQPAG(2) + 2) 150,160,170
      C
      C CURRENT CONTINUANT OF REQUESTED PAGE IF ALSO CURRENT PAGE
0068  150     REQPAG(2) = CURPAG(2)
0069          IF(REQPAG(1) .NE. CURPAG(1)) REQPAG(2) = 0
      C
      C BRING IN PROPER CONTINUANT
0071  155     CALL LVEXCH
0072  157     IF(IARG .GE. 2**SUFSZE) GO TO 200
0074          GO TO 120
      C
      C CONTINUANT NOT SPECIFIED, SET TO ZERO
0075  160     REQPAG(2) = 0
0076          GO TO 155
      C
      C CREATE NEW CONTINUANT
0077  170     CALL LVNCON
0078          GO TO 157
      C
      C TEST IFUNC AND RECONSTRUCT IF NECESSARY
0079  200     IF(IFUNC .GE. 2**SUFSZE) GO TO 300
0081          TEMPAG = REQPAG(1)
0082          REQPAG(1) = IFUNC
0083          IF(IFUNC .EQ. 0) REQPAG(1) = CURPAG(1)
0085          REQPAG(3) = REQPAG(1)
0086          CALL LVGRN(LNKSUF)
0087          IFUNC = LVIFSH(REQPAG(3),SUFSZE) .OR. LNKSUF
0088          REQPAG(1) = TEMPAG
      C
      C TEST THE SINK NODE AND RECONSTRUCT IF NECESSARY
      C   RANDOM NUMBER ?
0089  300     IF(ITYP1(1) .NE. 0) GO TO 400
0091          IF(IVALS(1) .GE. 2**SUFSZE) GO TO 400
0093          TEMPAG = REQPAG(1)
0094          IF(IVALS(1)) 310,320,330
      C
      C SINK NODE POINTS TO NEW PAGE
0095  310     REQPAG(4) = HACTPG(1) + 1
0096          GO TO 340
      C
      C SINK NODE POINTS TO CURRENT PAGE
0097  320     REQPAG(4) = CURPAG(1)
0098          GO TO 340
      C
      C SINK NODE POINTS TO DIFFERENT PAGE
0099  330     REQPAG(4) = IVALS(1)
0100  340     REQPAG(1) = REQPAG(4)
0101          CALL LVGRN(SNKSUF)
0102          REQPAG(1) = TEMPAG
```

```
      IVALS(1) = LVIFSH(REQPAG(4),SUFSZE) .OR. SNKSUF
C
C BEGIN SEARCH FOR EXISTING LIST
400   FULL = .FALSE.
      IF(INDXON .NE. 0) GO TO 420
C
C 'NORMAL' INSERTION, FIND BOTTOM OF LIST
410   IPOS = -1
420   TEMPOS = IPOS
      TMPREQ = REQPAG(2)
      CALL LVFDEX
      REQPAG(2) = TMPREQ
      IPOS = TEMPOS
C
C LIST FOUND ON REQUESTED CONTINUANT ?
      IF(ITESTR .GT. 0) GO TO 500
C
C REQUEST FOR NEW CONTINUANT (OR PAGE) ?
      IF(REQCON .EQ. -1) GO TO 500
C
C PLACE NEW LIST ACCORDING TO REQPAG(2) BUT FIRST SEARCH ELSEWHERE
      IPOS = TEMPOS
      TMPREQ = REQPAG(2)
      REQPAG(2) = -2
      CALL LVFDEX
      IPOS = TEMPOS
      REQPAG(2) = TMPREQ
C
C FOUND LIST ON DIFFERENT CONTINUANT ?
      IF(ITESTR .GT. 0) GO TO 450
C
C LIST DOES NOT EXIST ON ANY CONTINUANT
      IPOS = TEMPOS
      CALL LVFDEX
      IPOS = TEMPOS
      REQPAG(2) = TMPREQ
      GO TO 500
C
C IF CONT WAS NOT SPECIFIED, LIST FOUND ON NON-ZERO'TH CONTINUANT
C OTHERWISE, LIST WAS FOUND ON THE 'WRONG' CONTINUANT AND MVI MUST
C BE REORGANIZED AND PLACED ON REQCON
450   IF(REQCON .EQ. -2) GO TO 500
      REORG = .TRUE.
      CALL LVREOR(REQCON)
      REORG = .FALSE.
      GO TO 800
C
C PERFORM INSERTION
500   CALL LVNSRT
      IF(.NOT. FULL) GO TO 800
C
C CONTINUANT IS FULL, PLACE ON NEXT CONTINUANT IF SPACE IS AVAILABLE
      FULL = .FALSE.
C
C SPECIAL HANDLING FOR OVERFLOW ON INDEXED INSERTION
      IF(INDXON .NE. 0) GO TO 600
C
```

```
C DOES A PORTION OF THE MVI RESIDE ON THE CURRENT CONTINUANT ?
505   IF(ITESTR .LE. 0) GO TO 520
C
C SET MVI CONTINUATION FLAG
      FLGSPC(CTRL1+LASTLC)=FLGSPC(CTRL1+LASTLC) .OR. FLAG11
      NXTCON = .TRUE.
520   REQPAG(2) = REQPAG(2) + 1
      FLGSPC(CTRLPT+HDRFLG)=FLGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSD
     1 .OR. MWRITE
      CALL LVEXCH
      IF(MSARET .LE. 0) CALL LVNCON
      CALL LVFIND
      GO TO 500
C
C OVERFLOW ON INDEXED INSERTION
600   IF(IPOS .EQ. -1) GO TO 505
      CALL LVOVER
      GO TO 505
C
C RESET CONTINUANT USAGE RATIO
800   IF(RPLACE .EQ. .TRUE.) GO TO 820
      LNKSPC(CTRLPT+INSDEL) = LNKSPC(CTRLPT+INSDEL) + NVAL
C
C CONTINUANT HAS BEEN MODIFIED
820   FLGSPC(CTRLPT+HDRFLG)=FLGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSD
     1 .OR. MWRITE
      NODSPC(CTRLPT+REGAS) = REGASP
C
C IF LIST IS CONTAINED ON MORE THAN ONE CONTINUANT,
C SET BACK POINTING FLAG
      IF(NXTCON .EQ. .TRUE.)
     1 FLGSPC(CTRL1 + THIS) = FLGSPC(CTRL1 + THIS) .OR. FLAG10
C
C IF A SPECIFIC CONTINUANT WAS REQUESTED, SET REORG INHIBIT FLAG
      IF(REQCON .NE. -2)
     1 FLGSPC(CTRL1 + THIS) = FLGSPC(CTRL1 + THIS) .OR. FLAG12
      REQPAG(2) = -2
C
C CALL SECOND USER INSERTION STRATEGY ROUTINE ?
 1000 IF(IN2STR .EQ. .FALSE.) GO TO 1100
      IN1TMP = IN1STR
      IN2TMP = IN2STR
      IN1STR = .FALSE.
      IN2STR = .FALSE.
      DL1TMP = DL1STR
      DL2TMP = DL2STR
      DL1STR = .FALSE.
      DL2STR = .FALSE.
      CALL LVSTAC
      INSIDE = .TRUE.
      CALL USRIN2
      INSIDE = .FALSE.
      CALL LVPOP
      IN1STR = IN1TMP
      IN2STR = IN2TMP
      DL1STR = DL1TMP
      DL2STR = DL2TMP
C
C RESTORE FIND STRATEGY FLAGS
 1100 IF(INSIDE .EQ. .TRUE.) RETURN
      FD1STR = FD1TMP
      FD2STR = FD2TMP
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVNSRT
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1        DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,IN1TMP,
     2        DL2TMP,IN2TMP,FD2TMP,FULL,REORG,LSTCON,RPLACE
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR.
     1               INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2               LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1               HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2               DIRCNT,OUTLOC.OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF.MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK.FL5MSK.FLG67.
     1               FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2               FLAG15
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1               MSA,PAGLOC.CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1               INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1               USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR.IN2STR,FD1STR,FD2STR.
     1               DL1STR,DL2STR.IN2TMP,FD2TMP,DL2TMP.DUMPFL,
     2               FINDFL,DLETFL,NSRTFL
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1               LUN
      COMMON /LVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /LVFND/        COUNT,ABSPOS,LSTCON
      COMMON /LVINS1/ REORG.FULL,RPLACE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
      DATA TWO/"2/,THREE/"3/,NFLG67/"177774/.SVLRPL/"0/
C
C CALLS TO LVFINI OR LVFNV MUST PRECEDE A CALL TO THIS ROUTINE.
C
C INSERTION TYPE ?
C
D     PAUSE 'IN LVNSRT'
      RPLACE = .FALSE.
      IF(INDXON-1) 125,126,127
C
C IS THE GIRS BUFFER FULL ?
125   IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
C
C FORM FIRST WORD OF SINGLE OR MULTIVALUED FUNCTION
      FLGTMP = FL1MSK .OR. ITYP1(1)
      IF(NVAL .EQ. 1)GO TO 20
      LSTTMP = REGASP
      FLGTMP = FLGTMP .OR. FL0MSK .OR. FL2MSK
      GO TO 21
20    LSTTMP = IVALS(1)
C
```

```
C IF THIS FUNCTION ALREADY EXISTS, GO TO 43
21     IF(ITESTR .GT. 0) GO TO 43
C
C IF THAT ADDRESS IS ALREADY IN WORKING SPACE, GO TO 25
       IF((FL1MSK .AND. FLGSPC(CTRL1 + IADD)) .NE. 0) GO TO 25
C
C UPDATE REGASP(IF NECESSARY)
       IF(IADD .EQ. REGASP)REGASP = LSTSPC(CTRL1 + IADD)
C
C UPDATE AVAILABLE SPACE
       LSTSPC(CTRL1 + NODSPC(CTRL1 + IADD)) = LSTSPC(CTRL1 + IADD)
       NODSPC(CTRL1 + LSTSPC(CTRL1 + IADD)) = NODSPC(CTRL1 + IADD)
C
C INSERT FUNCTION
       NODSPC(CTRL1 + IADD) = IFUNC
       LSTSPC(CTRL1 + IADD) = LSTTMP
       LNKSPC(CTRL1 + IADD) = IADD
       FLGSPC(CTRL1 + IADD) = FLGSPC(CTRL1+IADD).OR.FLGTMP.OR.FL5MSK
C
C INSERT ANY ADDITIONAL FUNCTION VALUES
       IF(NVAL .EQ. 1) GO TO 100
       HEAD = IADD
       OLDLOC = IADD
       GO TO 50
C
C IF THAT ADDRESS CONTAINS THE HEAD OF A CONFLICT LIST, GO TO 60
25     IF((FL5MSK .AND. FLGSPC(CTRL1 + IADD)) .GT. 0) GO TO 60
C
C IF THAT ADDRESS CONTAINS A VALUE ON A MULTIVALUE LIST, GO TO 35
       IF(((FL2MSK .AND. FLGSPC(CTRL1 + IADD)) .GT. 0) .AND.
     1 (FL0MSK .AND. FLGSPC(CTRL1 + IADD)) .EQ. 0)          GO TO 35
C
C-----------------------------------------------------------------------------
C-THE ADDRESS CONTAINS A FUNCTION ON A CONFLICT LIST,BUT NOT THE HEAD OF LIST
       THIS = IADD
C
C FIND THE PRECEDING FUNCTION ON THE CONFLICT LIST
26     IF(LNKSPC(CTRL1 + LNKSPC(CTRL1 + THIS)) .EQ. IADD) GO TO 27
       THIS = LNKSPC(CTRL1 + THIS)
       GO TO 26
27     LAST = LNKSPC(CTRL1 + THIS)
       NEWLOC = REGASP
       IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
       CALL LVUPDT
C MOVE THE FUNCTION ON A CONFLICT LIST TO THE FIRST CELL OF AVAILABLE
C SPACE
       NODSPC(CTRL1 + NEWLOC) = NODSPC(CTRL1 + IADD)
       LSTSPC(CTRL1 + NEWLOC) = LSTSPC(CTRL1 + IADD)
       LNKSPC(CTRL1 + NEWLOC) = LNKSPC(CTRL1 + IADD)
       FLGSPC(CTRL1 + NEWLOC) = FLGSPC(CTRL1 + IADD) .OR. FL4MSK
       FLGSPC(CTRL1 + IADD) = 0
       LNKSPC(CTRL1 + LAST) = NEWLOC
C
C INSERT THIS FUNCTION AS THE HEAD OF A CONFLICT LIST
       NODSPC(CTRL1 + IADD) = IFUNC
```

```
      LNKSPC(CTRL1 + IADD) = IADD
      LSTSPC(CTRL1 + IADD) = LSTTMP
      FLGSPC(CTRL1 + IADD) =
     1   FLGSPC(CTRL1 + IADD) .OR. FLGTMP .OR. FL4MSK .OR. FL5MSK
      IF((FLGSPC(CTRL1 + NEWLOC) .AND. FL0MSK) .EQ. 0)  GO TO 34
C
C IF THE FUNCTION THAT WAS MOVED IS THE HEAD OF A MULTIVALUE LIST,FIX POINTERS
      NEXT = LSTSPC(CTRL1 + NEWLOC)
30    NEXT = LSTSPC(CTRL1 + NEXT)
      IF(LSTSPC(CTRL1 + NEXT) .NE. IADD) GO TO 30
      LSTSPC(CTRL1 + NEXT) = NEWLOC
C
C INSERT ANY ADDITIONAL FUNCTION VALUES
34    HEAD = IADD
      OLDLOC = IADD
      IF(NVAL .GT. 1) GO TO 50
      GO TO 100
C
C-----------------------------------------------------------------------------
C-THE ADDRESS CONTAINS A VALUE ON A MULTIVALUE LIST
35    NEWLOC = REGASP
      IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
      CALL LVUPDT
C
C MOVE THE VALUE ON A MULTIVALUE LIST TO THE FIRST CELL OF
C AVAILABLE SPACE
      NODSPC(CTRL1 + NEWLOC) = NODSPC(CTRL1 + IADD)
      LSTSPC(CTRL1 + NEWLOC) = LSTSPC(CTRL1 + IADD)
      LNKSPC(CTRL1 + NEWLOC) = LNKSPC(CTRL1 + IADD)
      FLGSPC(CTRL1 + NEWLOC) = FLGSPC(CTRL1 + IADD)
      FLGSPC(CTRL1 + IADD) = 0
C
C  RESET POINTERS
C
      L1 = LSTSPC(CTRL1 + NEWLOC)
      IF((FL0MSK .AND. FLGSPC(CTRL1 + L1)) .EQ. 0) GO TO 200
      LNKSPC(CTRL1 + LSTSPC(CTRL1 + L1)) = NEWLOC
      GO TO 201
200   LNKSPC(CTRL1 + L1) = NEWLOC
201   KZVAL = LSTSPC(CTRL1 + LNKSPC(CTRL1 + NEWLOC))
      IF((FLGSPC(CTRL1 + KZVAL) .AND. FL0MSK) .NE. 0)  GO TO 38
      LSTSPC(CTRL1 + LNKSPC(CTRL1 + NEWLOC)) = NEWLOC
      GO TO 39
38    LSTSPC(CTRL1 + KZVAL) = NEWLOC
39    NODSPC(CTRL1 + IADD) = IFUNC
C INSERT THIS FUNCTION AS THE HEAD OF A CONFLICT LIST
      LNKSPC(CTRL1 + IADD) = IADD
      LSTSPC(CTRL1 + IADD) = LSTTMP
      FLGSPC(CTRL1 + IADD) =
     1   FLGSPC(CTRL1 + IADD) .OR. FLGTMP .OR. FL4MSK .OR. FL5MSK
      GO TO 100
C
C-----------------------------------------------------------------------------
C-THE FUNCTION TO BE INSERTED IS ON THE CONFLICT LIST
43    HEAD = THIS
```

```
C
C IS THIS A SINGLE VALUE LIST OR MULTIVALUE LIST?
      IF(LSTHED .LT. 0) GO TO 51
C
C  OLDLOC IS THE LOCATION OF THE LAST VALUE ON THE MULTIVALUE LIST
C
      OLDLOC = LNKSPC(CTRL1 + LSTHED)
C
C-----------------------------------------------------------------------
C-INSERT ADDITIONAL FUNCTION VALUES
50    LSTASP = NODSPC(CTRL1 + REGASP)
      IN = 0
      GO TO 56
C
C-----------------------------------------------------------------------
C-FORM MULTIVALUE LIST TO ADD VALUE(S) TO SINGLE-VALUED FUNCTION
51    IN = 0
      IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP))GO TO 98
      LSTASP = NODSPC(CTRL1 + REGASP)
      NEWLOC = REGASP
      REGASP = LSTSPC(CTRL1 + REGASP)
      NODSPC(CTRL1 + NEWLOC) = LSTSPC(CTRL1 + THIS)
      TEMP = (FLGSPC(CTRL1 + THIS) .AND. FLG67)
      FLGSPC(CTRL1 + NEWLOC) = (TEMP .OR. FLGSPC(CTRL1 + NEWLOC))
      FLGSPC(CTRL1 + THIS) = (FLGSPC(CTRL1 + THIS) .AND. NFLG67)
      FLGSPC(CTRL1 + THIS) = (FL2MSK .OR. FLGSPC(CTRL1 + THIS))
      FLGSPC(CTRL1 + THIS) = (FL0MSK .OR. FLGSPC(CTRL1 + THIS))
      LNKSPC(CTRLPT+INSDEL) = LNKSPC(CTRLPT+INSDEL) + 1
      OLDLOC = THIS
C
C-----------------------------------------------------------------------
C INSERT ANOTHER VALUE ON MULTIVALUE LIST
52    FLGSPC(CTRL1 + NEWLOC) = (FL2MSK .OR. FLGSPC(CTRL1 + NEWLOC))
      FLGSPC(CTRL1 + NEWLOC) = (FL1MSK .OR. FLGSPC(CTRL1 + NEWLOC))
      LSTSPC(CTRL1 + OLDLOC) = NEWLOC
      LNKSPC(CTRL1 + NEWLOC) = OLDLOC
      OLDLOC = NEWLOC
56    NEWLOC = REGASP
      IF(IN .GT. 0) GO TO 57
C
C NO VALUES HAVE BEEN INSERTED YET
      IN = 1
      GO TO 58
C
C SOME VALUES HAVE BEEN INSERTED
57    IF(IN .EQ. NVAL) GO TO 67
      IN = IN+1
C
58    IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
      REGASP = LSTSPC(CTRL1 + REGASP)
      NODSPC(CTRL1 + NEWLOC) = IVALS(IN)
      FLGSPC(CTRL1 + NEWLOC) = (ITYP1(IN) .OR. FLGSPC(CTRL1 + NEWLOC))
      ITYP1(IN) = 0
      GO TO 52
C
C END MULTIVALUE LIST AND UPDATE AVAILABLE SPACE
67    LSTSPC(CTRL1 + OLDLOC) = HEAD
```

```
      NODSPC(CTRL1 + REGASP) = LSTASP
      LSTSPC(CTRL1 + LSTASP) = REGASP
      LNKSPC(CTRL1 + LSTSPC(CTRL1 + HEAD)) = OLDLOC
      GO TO 100
C
C-----------------------------------------------------------------------------
C-THE FUNCTION TO BE INSERTED IS NOT ON THE CONFLICT LIST
60    ASPREG = REGASP
      LSTASP = NODSPC(CTRL1 + REGASP)
      IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
C
C UPDATE AVAILABLE SPACE AND REGASP
      CALL LVUPDT
C
C INSERT FUNCTION IN FIRST CELL OF AVAILABLE SPACE
      NODSPC(CTRL1 + ASPREG) = IFUNC
      IF(NVAL .EQ. 1)GO TO 611
      LSTSPC(CTRL1 + ASPREG) = REGASP
      FLGSPC(CTRL1 + ASPREG) = (FL2MSK .OR. FLGSPC(CTRL1 + ASPREG))
      FLGSPC(CTRL1 + ASPREG) = (FL0MSK .OR. FLGSPC(CTRL1 + ASPREG))
      GO TO 612
611   LSTSPC(CTRL1 + ASPREG) = IVALS(1)
612   FLGSPC(CTRL1 + ASPREG) =
     1 FLGSPC(CTRL1 + ASPREG) .OR. ITYP1(1) .OR. FL1MSK .OR. FL4MSK
      LNKSPC(CTRL1 + ASPREG) = IADD
      LNKSPC(CTRL1 + LAST) = ASPREG
      IF(NVAL .EQ. 1) GO TO 100
C
C INSERT ADDITIONAL VALUES
      LSTASP = NODSPC(CTRL1 + REGASP)
      OLDLOC = ASPREG
      HEAD = ASPREG
      IN = 0
      GO TO 56
C
C  DESTRUCTIVE INSERTION
C
C A CALL TO LVFIND MUST PRECEDE A CALL TO EITHER 126 OR 127.
C GIVEN N VALUES OF TYPE K ON A LIST WHERE N.GE.0 , INDEXED
C INSERTIONS SHALL SUCCEED FOR IPOS.GE.1 .AND. IPOS .LE. N+1
C
C DEFEAT SAVED INDEX UNTIL NEXT RETRIEVAL.
126   FLGSPC(CTRL1 + THIS) = FLGSPC(CTRL1 + THIS) .OR. FL4MSK
      ABSPOS = IABS(IPOS)
      KPOS = IPOS
      INDEX = 0
C DOES THE IPOS'TH VALUE OF THE PROPER TYPE EXIST?
      IF(ITESTR .LT. 0) GO TO 90
C REPLACE VALUE AT LOCATION 'LOC'.  SVI OR MVI?
      RPLACE = .TRUE.
      IF(LSTHED .GT. 0) GO TO 356
C SVI
      LSTSPC(CTRL1 + LOC) = IVALS(1)
      SVIRPL = 1
      GO TO 365
C MVI
356   NODSPC(CTRL1 + LOC) = IVALS(1)
```

```
C REPLACE TYPE.
 365  FLGSPC(CTRL1 + LOC) =
     1    ((FLGSPC(CTRL1 + LOC) .AND. NFLG67) .OR. ITYP1(1))
      GO TO 100
C
C IPOS'TH VALUE WAS NOT FOUND, INDEXED INSERTION CAN STILL SUCCEED
C IF (IPOS-1) VALUE IS FOUND.  THIS THEN BECOMES A NORMAL INSERTION
C IF ABSPOS = 1 OR THE VALUE WILL BE THE LAST IN THE LIST.  OTHERWISE,
C THIS BECOMES A NONDESTRUCTIVE INSERTION TO THE FIRST POSITION IN
C THE LIST
C
 90   IF(ABSPOS .EQ. 1) GO TO 125
      IF(KPOS) 91,97.92
 91   KPOS = KPOS+1
      GO TO 93
 92   KPOS = KPOS-1
 93   CALL LVFIND
      IPOS = KPOS
      CALL LVFNV(INDEX,INDEX,INDEX,INDEX)
C FAILURE IF NO VALUE IS FOUND.
      IF(ITESTR .LT. 0) GO TO 97
C NORMAL INSERTION IF REQUEST WAS IPOS'TH FROM THE TOP.
      IF(KPOS .GT. 0) GO TO 125
C NONDESTRUCTIVE INSERTION AT THE BEGINNING OF THE LIST.
      NEWLOC = REGASP
      IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
      CALL LVUPDT
C SVI OR MVI?
      IF(LSTHED .GT. 0) GO TO 377
      GO TO 344
C
C  NONDESTRUCTIVE INSERTION
C
C IF IPOS = -1, PLACE AT THE END OF THE LIST (NORMAL INSERTION).
 127  IF(IPOS .EQ. -1) GO TO 125
C
C DEFEAT SAVED INDEX UNTIL NEXT RETRIEVAL.
      FLGSPC(CTRL1 + THIS) = FLGSPC(CTRL1 + THIS) .OR. FL4MSK
      ABSPOS = IABS(IPOS)
      KPOS = IPOS
      INDEX = 0
      NEWLOC = REGASP
C DOES THE IPOS'TH VALUE OF THE PROPER TYPE EXIST?
      IF(ITESTR .LT. 0) GO TO 90
      IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
      CALL LVUPDT
C SVI OR MVI?
      IF(LSTHED .LT. 0) GO TO 344
C MVI
      IF(KPOS .LT. 0) GO TO 347
C
C PLACE VALUE AT THE IPOS'TH POSITION (WRT ITYP) FROM THE TOP OF LIST
 377  ISTLOC = LNKSPC(CTRL1 + LOC)
      NODSPC(CTRL1 + NEWLOC) = IVALS(1)
      LSTSPC(CTRL1 + NEWLOC) = LOC
      LNKSPC(CTRL1 + NEWLOC) = ISTLOC
      FLGSPC(CTRL1 + NEWLOC) = FL1MSK .OR. FL2MSK .OR. ITYP1(1)
      IF(LOC .NE. LSTHED) GO TO 321
      LSTSPC(CTRL1 + LSTSPC(CTRL1 + ISTLOC)) = NEWLOC
      GO TO 322
321   LSTSPC(CTRL1 + ISTLOC) = NEWLOC
322   LNKSPC(CTRL1 + LOC) = NEWLOC
      GO TO 100
C
C PLACE VALUE AT THE IPOS'TH POSITION (WRT ITYP) FROM THE BOTTOM OF
C THE LIST
347   NODSPC(CTRL1 + NEWLOC) = IVALS(1)
      LSTSPC(CTRL1 + NEWLOC) = LSTSPC(CTRL1 + LOC)
      LNKSPC(CTRL1 + NEWLOC) = LOC
      FLGSPC(CTRL1 + NEWLOC) = FL1MSK .OR. FL2MSK .OR. ITYP1(1)
      IF((FLGSPC(CTRL1 + LSTSPC(CTRL1 + LOC)) .AND. FL0MSK) .EQ. 0)
     1    GO TO 323
      KZVAL = LSTSPC(CTRL1 + LOC)
      LNKSPC(CTRL1 + LSTSPC(CTRL1 + KZVAL)) = NEWLOC
      GO TO 324
323   LNKSPC(CTRL1 + LSTSPC(CTRL1 + LOC)) = NEWLOC
324   LSTSPC(CTRL1 + LOC) = NEWLOC
      GO TO 100
C
C CREATE MVI WITH NEW VALUE AT THE TOP OF THE LIST.
344   IF(REGASP .EQ. LSTSPC(CTRL1 + REGASP)) GO TO 98
      NWLOC2 = REGASP
      CALL LVUPDT
      NODSPC(CTRL1 + NEWLOC) = IVALS(1)
      LSTSPC(CTRL1 + NEWLOC) = NWLOC2
      LNKSPC(CTRL1 + NEWLOC) = NWLOC2
      FLGSPC(CTRL1 + NEWLOC) = FL1MSK .OR. FL2MSK .OR. ITYP1(1)
      NODSPC(CTRL1 + NWLOC2) = LSTSPC(CTRL1 + THIS)
      LSTSPC(CTRL1 + NWLOC2) = THIS
      LNKSPC(CTRL1 + NWLOC2) = NEWLOC
      KLGTEP = FLGSPC(CTRL1 + THIS) .AND. FLG67
      FLGSPC(CTRL1 + NWLOC2) = (FL1MSK .OR. FL2MSK) .OR. KLGTEP
      LSTSPC(CTRL1 + THIS) = NEWLOC
      FLGSPC(CTRL1 + THIS) =
     1    (FLGSPC(CTRL1 + THIS) .OR. FL0MSK) .OR. FL2MSK
C FLAG 4 IS SET BECAUSE THIS INSERTION MIGHT BE A RECREATION OF AN
C OLD LIST
100   FLGSPC(CTRL1 + THIS) = FLGSPC(CTRL1 + THIS) .OR. FL4MSK
      IVAL = IVALS(1)
C "FAILURE" IF IFUNC+IARG DID NOT PREVIOUSLY EXIST
C     IF(((FLGSPC(CTRL1 + THIS) .AND. FL0MSK) .NE. 0) .OR.
C    1    SVIRPL .EQ. 1) ITESTR  = 1
97    IPOS = 1
      SVIRPL = 0
      ITYP = 3
      INDXON = 0
      NVAL = 1
      ITYP1(1) = 0
      RETURN
C
C CONTINUANT IS FULL, TRY AGAIN
98    FULL = .TRUE.
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVUPDT
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1              FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2              FLAG15
C
C THIS ROUTINE UPDATES AVAILABLE SPACE AND THE REGISTER OF AVAILABLE
C SPACE - REGASP
C
D     PAUSE 'IN LVUPDT'
      LSTSPC(CTRL1 + NODSPC(CTRL1 + REGASP)) = LSTSPC(CTRL1 + REGASP)
      NODSPC(CTRL1 + LSTSPC(CTRL1 + REGASP)) = NODSPC(CTRL1 + REGASP)
      REGASP = LSTSPC(CTRL1 + REGASP)
      XXX=1000
      IF((FLGSPC(CTRL1+REGASP).OR.FL3MSK).NE.FL3MSK)XXX=XXX*XXX
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVDLEX
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1        DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,FD1TMP,
     2        DL2TMP,IN2TMP,FD2TMP,INSIDE,REORG,FULL,LSTCON,RPLACE,
     3        BAKCON
      COMMON /IVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1               INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2               LNKSUF,SNKSUF,INSTYP
      COMMON /IVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1               HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2               DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /IVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /IVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1               FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2               FLAG15
      COMMON /IVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1               MSA,PAGLOC,CURENT
      COMMON /IVBUFR/ PAGSZE,NWCHAN,OLCHAN CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1               INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /IVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1               USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /IVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1               DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2               FINDFL,DLETFL,NSRTFL
      COMMON /IVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1               LUN
      COMMON /IVFND/        COUNT,ABSPOS,LSTCON
      COMMON /IVINS1/ REORG,FULL,RPLACE
      COMMON /IVDEL1/ NUMRET,BAKCON
      COMMON /IVVTR1/ NODSPC(1)
     1    /IVVTR2/ LSTSPC(1)
     2    /IVVTR3/ LNKSPC(1)
     3    /IVVTR4/ FLGSPC(1)
C
      DATA INSIDE /.FALSE./
C
C THE DELETE EXECUTIVE ROUTINE OBTAINS THE CORRECT P,C FOR SUBROUTINE
C LVDLET TO OPERATE ON.
C
D     PAUSE 'IN LVDLEX'
C IS LVDLET BEING CALLED FROM A DELETE STRATEGY ROUTINE ?
      IF(INSIDE .EQ. .TRUE.) GO TO 100
C
C TO PREVENT RECURSION, SAVE THE FIND STRATEGY FLAGS AND TURN THEM OFF
      FD1TMP = FD1STR
      FD2TMP = FD2STR
      FD1STR = .FALSE.
      FD2STR = .FALSE.
C
C CALL USER'S FIRST DELETE STRATEGY ROUTINE ?
      IF(DL1STR .EQ. .FALSE.) GO TO 100
C
C TO PREVENT RECURSION, INHIBIT CALLS TO ALL USER STRATEGY ROUTINES
```

```
         DL1TMP = DL1STR
         DL2TMP = DL2STR
         DL1STR = .FALSE.
         DL2STR = .FALSE.
         IN1TMP = IN1STR
         IN2TMP = IN2STR
         IN1STR = .FALSE.
         IN2STR = .FALSE.
C
C SET UP FOR FIRST USER ROUTINE
         CALL LVSTAC
         INSIDE = .TRUE.
         CALL USRDL1
         INSIDE = .FALSE.
         CALL LVPOP
         DL1STR = DL1TMP
         DL2STR = DL2TMP
         IN1STR = IN1TMP
         IN2STR = IN2TMP
C
C PROCEED WITH DELETION ?
         IF(DLETFI .EQ. .FALSE.) GO TO 600
C
C BRING IN PROPER CONTINUANT
100      J = 0
         CALL LVFDEX(J,J,J,J,J)
C
C NO LIST TO BE DELETED ?
         IF(ITESTR .IT. 0) GO TO 600
C
C ASSUME LIST DOES NOT PROCEED TO ANOTHER CONTINUANT
200      LSTCON = .FALSE.
         BAKCON = .FALSE.
C
C NUMRET COUNTS THE NUMBER OF LOCATIONS RETURNED TO AVAILABIE SPACE
         NUMRET = 0
         CALL LVDLET
C
C UPDATE CONTINUANT FILL QUANTITY
         LNKSPC(CTRIPT+INSDEL) = LNKSPC(CTRIPT+INSDEL) - NUMRET
C
C CONTINUANT HAS BEEN MODIFIED
         FLGSPC(CTRIPT+HDRFLG) = FLGSPC(CTRIPT+HDRFLG) .OR. MWRITE
C
C INDEXED DELETE ?
         IF(INDXON .EQ. 1) GO TO 400
C
C FINISHED ?
         IF(LSTCON .EQ. .FALSE.) GO TO 600
C
C EXAMINE NEXT CONTINUANT
300      REQPAG(2) = CURPAG(2) + 1
         J = 0
         CALL LVFDEX(J,J,J,J,J)
C
C NO MORE CONTINUANTS ?
         IF(MSARET .IE. 0) GO TO 500
```

AD-A099 125 DAVID W TAYLOR NAVAL SHIP RESEARCH AND DEVELOPMENT CE--ETC F/G 5/2
PAGED GIRS (GRAPH INFORMATION RETRIEVAL SYSTEM) USERS MANUAL.(U)
MAY 81 I S ZARITSKY
UNCLASSIFIED DTNSRDC-81/024 NL

2 OF 2
AD A
099 125

END
DATE
FILMED
6-81
DTIC

```
C
C DOES A PORTION OF THE LIST RESIDE ON CURRENT [illegible] ?
         IF(ITESTR .LT. 0) GO TO 300
         GO TO 200
C
C RESET DEFAULT TO "DELETE ENTIRE LIST"
400      INDXON = 0
         IF(BAKCON .EQ. .FALSE.) GO TO 450
C
C LIST NO LONGER POINTS FORWARD TO A FOLLOWING CONTINUANT, REMOVE FLAG
         IPOS = -1
         CALL LVFDEX
         FLGSPC(CTRL1+LOC) = FLGSPC(CTRL1+LOC) .AND. .NOT. FLAG11
450      IF(LSTCON .EQ. .FALSE.) GO TO 600
C
C LIST NO LONGER POINTS BACKWARD TO A PREVIOUS CONTINUANT, REMOVE FLAG
         IPOS = 1
         CALL LVFDEX
         FLGSPC(CTRL1+THIS) = FLGSPC(CTRL1+THIS) .AND. .NOT. FLAG10
         GO TO 600
C
C ERROR, LIST CONTINUATION FLAG BUT NO MORE CONTINUANTS !
500      ERRNUM = 50
         MODE = BCD
         PAGES = -1
         DUMP = 0
         CALL LVDUMP(DUMP)
         X = 1000
         X = X * X
         STOP
C
C CALL SECOND USER DELETION STRATEGY ROUTINE ?
600      IF(DL2STR .EQ. .FALSE.) GO TO 700
         DL1TMP = DL1STR
         DL2TMP = DL2STR
         DL1STR = .FALSE.
         DL2STR = .FALSE.
         IN1TMP = IN1STR
         IN2TMP = IN2STR
         IN1STR = .FALSE.
         IN2STR = .FALSE.
         CALL LVSTAC
         INSIDE = .TRUE.
         CALL USRDL2
         INSIDE = .FALSE.
         CALL LVPOP
         DL1STR = DL1TMP
         DL2STR = DL2TMP
         IN1STR = IN1TMP
         IN2STR = IN2TMP
C
C RESTORE FIND STRATEGY FLAGS
700      IF(INSIDE .EQ. .TRUE.) RETURN
         FD1STR = FD1TMP
         FD2STR = FD2TMP
         RETURN
         END
```

```
C
C
C
      SUBROUTINE LVDLET
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1      DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,DL1TMP,
     2      DL2TMP,IN2TMP,FD2TMP,REORG,FULL,LSTCON,RPLACE,BAKCON
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1              INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2              LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LS[illegible]PG(4),MSARPT,
     1              HREQPG,NXTMSA,HACTPG(2),[illegible]CT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1              FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2              FLAG15
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2              FINDFL,DLETFL,NSRTFL
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1              LUN
      COMMON /LVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /LVSTAK/ CURLEV,NUMVAR,STACK(1)
      COMMON /LVFND/       COUNT,ABSPOS,LSTCON
      COMMON /LVINS1/ REORG,FULL,RPLACE
      COMMON /LVDEL1/ NUMRET,BAKCON
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
      DATA NFL023/"177517/
C
C
C
C DOES THE LIST EXIST ?
D     PAUSE 'IN LVDLET'
      IF(ITESTR .LT. 0) RETURN
C
C SVL OR MVL ?
      IF(LSTHED .LT. 0) GO TO 200
C
C INDEXED DELETE ?
      IF(INDXON .EQ. 1) GO TO 500
C
C DELETE ENTIRE MULTIVALUE LIST
      ISADD = LSTHED
      LOC = THIS
 100  NXTADD = LSTSPC(CTRL1 + ISADD)
      IF((FLGSPC(CTRL1 + NXTADD) .AND. FLAG11) .NE. 0) LSTCON = .TRUE.
      NODSPC(CTRL1 + ISADD) = NODSPC(CTRL1 + REGASP)
      LSTSPC(CTRL1 + ISADD) = REGASP
      LNKSPC(CTRL1 + ISADD) = 0
      FLGSPC(CTRL1 + ISADD) = 0
      LSTSPC(CTRL1 + NODSPC(CTRL1 + REGASP)) = ISADD
      NODSPC(CTRL1 + REGASP) = ISADD
      NUMRET = NUMRET + 1
      IF((FLGSPC(CTRL1 + NXTADD) .AND. FL0MSK) .NE. 0) GO TO 200
      ISADD = NXTADD
      GO TO 100
C
C DELETE SINGLE VALUED FUNCTION
C FORWARD OR BACK MVI CONTINUANT POINTER FLAGS MAY HAVE TO BE REMOVED
200   IF((FLGSPC(CTRL1 + THIS) .AND. FLAG11) .NE. 0) LSTCON = .TRUE.
      IF((FLGSPC(CTRL1 + THIS) .AND. FLAG10) .NE. 0) BAKCON = .TRUE.
C IF THE LIST EXTENDED TO BOTH A PREVIOUS AND FOLLOWING CONTINUANT,
C DO NOT REMOVE POINTER FLAGS
      IF(LSTCON .EQ. .FALSE. .OR. BAKCON .EQ. .FALSE.) GO TO 220
      LSTCON = .FALSE.
      BAKCON = .FALSE.
C IS THE FUNCTION HEAD OF A CONFLICT LIST
220   IF(THIS .NE. IADD) GO TO 400
      NXFUNC = LNKSPC(CTRL1 + IADD)
C IF THIS FUNCTION IS THE ONLY ONE ON THE CONFLICT LIST, GO TO 300.
C OTHERWISE, PLACE NEXT FUNCTION ON CONFLICT LIST IN 'HEAD OF
C CONFLICT LIST' LOCATION (IADD)
      IF(NXFUNC .EQ. IADD) GO TO 300
      NODSPC(CTRL1 + IADD) = NODSPC(CTRL1 + NXFUNC)
      LSTSPC(CTRL1 + IADD) = LSTSPC(CTRL1 + NXFUNC)
      LNKSPC(CTRL1 + IADD) = LNKSPC(CTRL1 + NXFUNC)
      FLGSPC(CTRL1 + IADD) = FLGSPC(CTRL1 + NXFUNC)
      FLGSPC(CTRL1 + IADD) = FLGSPC(CTRL1 + IADD) .OR. FL5MSK
      IF((FLGSPC(CTRL1 + IADD) .AND. FL6MSK) .EQ. 0) GO TO 270
C IF THE MOVED FUNCTION IS A MVI, THE POINTER FROM THE LAST VALUE OF
C THE LIST TO THE HEAD MUST BE UPDATED.
      KVAL = LSTSPC(CTRL1 + IADD)
250   KVAL = LSTSPC(CTRL1 + KVAL)
      IF((FLGSPC(CTRL1 + LSTSPC(CTRL1 + KVAL)) .AND. FL0MSK) .EQ. 0)
     1 GO TO 250
      LSTSPC(CTRL1 + KVAL) = IADD
270   LOC = NXFUNC
C RETURN LOCATION TO AVAILABLE SPACE
300   NODSPC(CTRL1 + LOC) = NODSPC(CTRL1 + REGASP)
      LSTSPC(CTRL1 + LOC) = REGASP
      LNKSPC(CTRL1 + LOC) = 0
      FLGSPC(CTRL1 + LOC) = 0
      NODSPC(CTRL1 + LSTSPC(CTRL1 + LOC)) = LOC
      LSTSPC(CTRL1 + NODSPC(CTRL1 + LOC)) = LOC
      NUMRET = NUMRET + 1
      RETURN
C
C FUNCTION TO BE DELETED IS NOT THE HEAD OF A CONFLICT LIST.
C THE FUNCTION PRECEDING THIS (FUNCTION BEING DELETED) MUST POINT TO
C THE FUNCTION FOLLOWING THIS
400   LNKSPC(CTRL1 + LAST) = LNKSPC(CTRL1 + THIS)
      GO TO 300
C
```

```
C *** INDEXED DELETE
C
C FUNCTION MUST BE A MVI  OR, IF SVL, ABS(IPOS) = 1 WITH PROPER TYPE.
C DELETE VALUE AT LOC.  DEFEAT SAVED INDEX FOR THIS LIST UNTIL AFTER
C NEXT RETRIEVAL.
500   FLGSPC(CTRL1 + THIS) = FLGSPC(CTRL1 + THIS) .OR. FL4MSK
C
C INDEXED DELETE CAN BE REDUCED TO FOUR CASES.  DELETE VALUE IN
C FIRST, MIDDLE, OR LAST POSITION ON LIST, OR REDUCE TO SVL.
C
      NEXT = LSTSPC(CTRL1 + LOC)
      NPRIOR = LNKSPC(CTRL1 + LOC)
C
C IS LOC THE  LAST POSITION IN THE LIST ?
      IF(NEXT .EQ. THIS) GO TO 600
C
C IS LOC THE FIRST POSITION IN THE LIST ?
      IF(LSTSPC(CTRL1 + NPRIOR) .EQ. THIS) GO TO 700
C
C VALUE IS IN A MIDDLE POSITION IN THE LIST.  RECONNECT VALUES
C PRECEEING AND FOLLOWING LOC.
C
      LSTSPC(CTRL1 + NPRIOR) = NEXT
      LNKSPC(CTRL1 + NEXT) = NPRIOR
      GO TO 300
C
C DELETE VALUE IN LAST POSITION IN LIST
600   LSTSPC(CTRL1 + NPRIOR) = NEXT
      NEXT1 = LSTSPC(CTRL1 + NEXT)
      LNKSPC(CTRL1 + NEXT1) = NPRIOR
      IF((FLGSPC(CTRL1 + LOC) .AND. FLAG11) .NE. 0)
     1 FLGSPC(CTRL1+NPRIOR) = FLGSPC(CTRL1+NPRIOR) .OR. FLAG11
      GO TO 800
C
C DELETE VALUE IN FIRST POSITION IN LIST
700   LNKSPC(CTRL1 + NEXT) = NPRIOR
      LSTSPC(CTRL1 + THIS) = NEXT
C
C CONVERT TO A SINGLE VALUE LIST ?
C
800   IF(LNKSPC(CTRL1 + NPRIOR) .NE. NPRIOR) GO TO 300
C IF DELETING LAST VALUE, RESET NEXT TO FIRST VALUE
      IF(NEXT .EQ. THIS) NEXT = NPRIOR
      LSTSPC(CTRL1 + THIS) = NODSPC(CTRL1 + NEXT)
      FLGSPC(CTRL1 + THIS) =
     1 (FLGSPC(CTRL1 + THIS) .OR. FLGSPC(CTRL1 + NEXT)) .AND. NFL023
      FLGSPC(CTRL1 + NEXT) = 0
      LNKSPC(CTRL1 + NEXT) = 0
      NODSPC(CTRL1 + NEXT) = NODSPC(CTRL1 + REGASP)
      LSTSPC(CTRL1 + NEXT) = REGASP
      NODSPC(CTRL1 + LSTSPC(CTRL1 + NEXT)) = NEXT
      LSTSPC(CTRL1 + NODSPC(CTRL1 + NEXT)) = NEXT
      NUMRET = NUMRET + 1
      GO TO 300
      END
```

```
C
C
C
      SUBROUTINE LVSETP
      IMPLICIT INTEGER(A-Z)
      REAL*4 DEFEXT,CORE,TOP,BOTTOM
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1      DL2STR,DUMPFL,CURENT,IN2TMP,FD2TMP,DL2TMP,FINDFL,
     2      DLETFL,NSRTFL,REORG,FULL,RPLACE
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1              INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2              LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1              FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2              FLAG15
      COMMON /LVRAND/ PRIME,SEED,NROW,DNODE,DROW,OLDNOD,LISTSZ,
     1              GRNTBL(256)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2              FINDFL,DLETFL,NSRTFL
      COMMON /LVVSEQ/ ISEQSZ,ISOPOS,LASTSO,SEQSPC(1)
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY RCD,MODE,PAGES,
     1              LUN
      COMMON /LVSTAK/ CURLEV,NUMVAR,STACK(1)
      COMMON /LVUTIL/ FILSPC(39),DEFEXT(2)
      COMMON /LVINS1/ REORG,FULL,RPLACE
      COMMON /LVRUN/        RUNTYP,CORE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
D     PAUSE 'IN LVSETP'
      IF(SNGLBK) GO TO 120
      PAGHDR = PAGSZE + HDRSZE
      BLKSZE = PAGHDR/64
      PAGHD4 = 4*PAGHDR
      DIRSZE = 64*((INCORE/64) + 1)
      DIRBLK = DIRSZE/64
      BUFSZE = DIRSZE + (INCORE*PAGHDR)
C
      TYPE 1
1     FORMAT(' PLEASE ENTER FILE NAMES OF OLD AND NEW GRAPHS'/
     1      ' IN COMMAND STRING FORMAT (NEW.EXT = OLD.EXT)'/
     2      ' .GRF IS ASSUMED EXTENSION'/)
      IF(ICSI(FILSPC,DEFEXT,,,0) .NE. 0) STOP 'INVALID COMMAND STRING'
C     RUN TYPE 1 - CREATE NEW GRAPH
```

```
C     RUN TYPE 2 - UPDATE OLD GRAPH
C     RUN TYPE 3 - QUERY  OLD GRAPH
      IF(RUNTYP .EQ. 3) GO TO 100
C     ASSIGN CHANNEL TO OUTPUT (NEW) GRAPH
      NWCHAN = IGETC()
      IF(NWCHAN .LT. 0) STOP 'NO OUTPUT CHANNEL AVAILABLE'
      IF(IFETCH(FILSPC(1)) .NE. 0) STOP 'OUTPUT DEVICE HANDLER FETCH
     1 FAILURE'
      IF(IENTER(NWCHAN,FILSPC(1),0) .LT. 0) STOP 'ENTRY FAILURE'
      IF(RUNTYP .EQ. 1) GO TO 110
C     ASSIGN CHANNEL TO INPUT (OLD) GRAPH
100   OLCHAN = IGETC()
      IF(OLCHAN .LT.0) STOP 'NO INPUT CHANNEL AVAILABLE'
      IF(IFETCH(FILSPC(16)) .NE. 0) STOP 'INPUT DEVICE HANDLER FETCH
     1 FAILURE'
      IF(LOOKUP(OLCHAN,FILSPC(16)) .LT. 0) STOP 'INPUT FILE LOOKUP
     1 FAILURE'
C     READ OLD GRAPH INTO BUFFER AS DEFINED BY STORED IN-CORE DIRECTORY
      CALL LVFECH
D     PAUSE ' LEAVING LVSETP 1'
      RETURN
C
C CREATION RUN
110   READCT = 1
      USECNT = 1
C
120   SEED = PRIME/2
      NROW = SEED
      OLDNOD = SEED - PRIME
      DROW = PRIME
      DNODE = PRIME
      LISTSZ = 1
      REGASP = 1
C SET UP SINGLE BLOCK?
      IF(SNGLBK) GO TO 160
140    DO 145 I = 1,64
       J = 4*(I-1)
       GRNTBL(J + OLDNDH) = OLDNOD
       GRNTBL(J + DNODEH) = DNODE
       GRNTBL(J + NROWH)  = NROW
145    GRNTBL(J + DROWH)  = DROW
      TOP = INCORE-1
      BOTTOM = INCORE
      IF(BOTTOM .EQ. 0) BOTTOM = 1
      CORE = TOP/BOTTOM
C SET UP DIRECTORY AVAILABLE SPACE
      DREGSP = 1
       DO 150 I = 2,DIRSZE
       NODSPC(I) = I-1
       LSTSPC(I-1) = I
       LNKSPC(I) = 0
150    FLGSPC(I) = FL3MSK
      NODSPC(1) = DIRSZE
      LNKSPC(1) = 0
      FLGSPC(1) = FL3MSK
      LSTSPC(DIRSZE) = 1
C
```

```
C SET UP WRKSPC OR SINGLE C. PT.
      CTRLPT = DIRSZE
      ISTLOC = DIRSZE + HDRSZE
160   ENDLOC = CTRLPT + PAGHDR
      CTRL1  = CTRLPT + HDRSZE
      CNTRL1 = CTRL1       + 1
       DO 170 I = CNTRL1,ENDLOC
       LNKSPC(I) = 0
170   FLGSPC(I) = FL3MSK
C
C INITIALIZE AVAILABLE SPACE RING STRUCTURE OF THE REQUESTED CONTROL
C POINT OF WRKSPC.  IF THIS IS AT THE BEGINNING OF A CREATION RUN,
C IT IS C. PT. #1, THEN COPY TO THE OTHER C. PTS.
      SETUP = .TRUE.
      CALL LVGRN(REGASP)
      OLD=REGASP
        DO 180 I=2,PAGSZE
        CALL LVGRN(NEW)
        NODSPC(NEW + CTRL1) =OLD
        LSTSPC(OLD + CTRL1) =NEW
180     OLD=NEW
      NODSPC(REGASP + CTRL1) =OLD
      LSTSPC(OLD + CTRL1) =REGASP
      NROW=SEED
      OLDNOD = SEED - PRIME
      DROW = PRIME
      DNODE=PRIME
      SETUP = .FALSE.
      LIST = INCORE
      IF(SNGLBK) LIST = 1
        DO 200 I=1,LIST
C
C SET HEADER WORDS FOR THE CONTINUANTS
        NODSPC(CTRLPT+REGAS)=REGASP
        LNKSPC(CTRLPT+INSDEL)=0
        LNKSPC(CTRLPT+USECT)=0
        FLGSPC(CTRLPT+HDRFLG) = 0
        FLGSPC(CTRLPT+READVL)=READCT
        IF(SNGLBK) GO TO 200
        IF(I .EQ. 1) GO TO 195
C COPY "AS" RING STRUCTURE TO REMAINING C. PTS.
      DO 190 K=1,PAGSZE
      NODSPC(CTRL1 + K) = NODSPC(ISTLOC + K)
      LSTSPC(CTRL1 + K) = LSTSPC(ISTLOC + K)
      LNKSPC(CTRL1 + K) = 0
190   FLGSPC(CTRL1 + K) = FL3MSK
195     CTRLPT=CTRLPT+PAGHDR
        CTRL1=CTRLPT+HDRSZE
200     CONTINUE
      IF(.NOT. SNGLBK) GO TO 210
C
C COMPLETE HEADER WORDS FOR SINGLE CONTINUANT
C   ASSUME CORRECT OUTCORE DIRECTORY BLOCK IS IN CORE
      NODSPC(CTRLPT + THSMSA) = NXTMSA
      LSTSPC(CTRLPT + PAGENO) = REQPAG(1)
      LSTSPC(CTRLPT + CONTNO) = REQPAG(2)
      NXTMSA = NXTMSA + BLKSZE
```

```
D     PAUSE ' LEAVING LVSETP 2'
      RETURN
C
C CREATE CONTINUANTS BY PAGE ORDER
 210  PAGNUM = 0
      CTRLPT = DIRSZE
      CTRL1  = CTRLPT + HDRSZE
C
C CREATE PAGE PAGNUM, CONTINUANT CONTIN.
C COMPLETE HEADER WORDS FOR CONTINUANTS
      LIST = HREQPG
      DIRCNT = 0
      DIRPAG = 1
C INITIALIZE NXTMSA TO LOCATION OF PAGE 1, CONT 0
      NXTMSA = DIRBLK + 18
        DO 300 I=1,LIST
        PAGNUM = PAGNUM+1
        NUMCON = STACK(PAGNUM) + 1
        DIRCNT = DIRCNT + 1
        IF(DIRCNT .NE. 1) GO TO 215
      DO 214 J=1,256
 214  OUTDIR(J) = 0
C
 215  DO 220 K=1,NUMCON
      CONTIN = K-1
      NODSPC(CTRLPT + THSMSA) = NXTMSA
      LSTSPC(CTRLPT + PAGENO) = PAGNUM
      LSTSPC(CTRLPT + CONTNO) = CONTIN
C OUTPUT UNUSED CONTINUANT TO DISK
      LENGTH = PAGHDR
      BUFLOC = CTRLPT + 1
      ERRNUM = 1
      MSA = NODSPC(CTRLPT + THSMSA)
      CALL LVPAGW
C ENTER CONTINUANT LOCATION INTO OUTCORE DIRECTORY
      OUTLOC = 1 + 64*(DIRCNT-1) + CONTIN
      OUTDIR(OUTLOC) = MSA
      NXTMSA = NXTMSA + BLKSZE
C UPDATE CONTROL POINTER
      CTRLPT=CTRLPT+PAGHDR
      IF(CTRLPT.GE.BUFSZE) CTRLPT=DIRSZE
 220  CONTINUE
C SAVE THIS BLOCK OF THE OUTCORE DIRECTORY IF ALL 4 SEGMENTS ARE FILLED
        IF(DIRCNT .LT. 4) GO TO 300
        CALL LVDRWR
        DIRCNT = 0
        DIRPAG = DIRPAG + 1
 300    CONTINUE
C
C SAVE MOST RECENT OUTCORE DIRECTORY BLOCK IF NECESSARY
      IF(DIRCNT .EQ. 0) GO TO 310
      CALL LVDRWR
C
C ZERO OUT REMAINING UNUSED OUTCORE DIRECTORY BLOCKS
 310    DO 312 I=1,256
 312    OUTDIR(I) = 0
C
```

```
315   IF(DIRPAG .EQ. 16) GO TO 320
      DIRPAG = DIRPAG + 1
      CALL LVDRWR
      GO TO 315
C
C BRING FIRST OUTCORE DIRECTORY BLOCK INTO MAIN MEMORY
320   REQPAG(1) = 1
      REQPAG(2) = 0
      CALL LVMSA(DUMMY)
C
C IF IT IS NOT THERE, BRING PAGE 1, CONTINUANT 0 BACK INTO CORE
      CTRLPT = DIRSZE
      CTRL1 = CTRLPT + HDRSZE
      IF((LSTSPC(CTRLPT + PAGENO) .EQ. 1) .AND.
     1  (LSTSPC(CTRLPT + CONTNO) .EQ. 0)) GO TO 340
      MSA = OUTDIR(1)
      BUFLOC = CTRLPT + 1
      LENGTH=PAGHDR
      ERRNUM = 2
      CHAN = NWCHAN
      CALL LVPAGR(CHAN)
C
C INSERT INCORE CONTINUANTS INTO DIRECTORY PAGE
340   CTRLPT = - HDRSZE
      CTRL1 = 0
      TMPSZE = PAGSZE
      PAGSZE = DIRSZE
      CNTRL1 = DIRSZE - PAGHDR
      INDXON = 0
      NVAL=1
      INSTYP=1
        DO 400 K=1,INCORE
        CNTRL1 = CNTRL1 + PAGHDR
        PAGE = LSTSPC(CNTRL1 + PAGENO)
        CONT = LSTSPC(CNTRL1 + CONTNO)
        IARG = CONT + 1
        IFUNC= PAGE
        SRCSUF = IARG
        LNKSUF = IFUNC
D     PAUSE ' B FIND'
        CALL LVFIND
D     PAUSE ' AFT FIND'
        IVALS(1)= CNTRL1
        ITYP1(1)=1
        SNKSUF = IVALS(1)
        CALL LVNSRT
400     CONTINUE
      DREGSP=REGASP
      REGASP=1
      PAGSZE = TMPSZE
C
C ESTABLISH REGISTERS
C    PAGE 1, CONT 0 IS:
C     HIGHEST ACTIVE PAGE
C     CURRENT PAGE - CONT
C     PREVIOUS CURRENT PAGE - CONT
C     REQUESTED PAGE - CONT
```

```
C
      HACTPG(1) = 0
      HACTPG(2) = OUTDIR(1)
      CTRLPT = DIRSZE
      CTRL1  = CTRLPT + HDRSZE
C
      CURPAG(1) = 1
      CURPAG(2) = 0
      CURPAG(3) = OUTDIR(1)
      CURPAG(4) = CTRLPT
C
      REQPAG(1) = CURPAG(1)
      REQPAG(2) = -2
      REQPAG(3) = CURPAG(3)
      REQPAG(4) = CURPAG(4)
      CURENT = .TRUE.
C
C DECLARE THE LAST CONTROL POINT AS AVAILABLE.
      LEASTV = FREE
      CNTRL1 = BUFSZE - PAGHDR
      LSTVPG(1) = LSTSPC(CNTRL1 + PAGENO)
      LSTVPG(2) = LSTSPC(CNTRL1 + CONTNO)
      LSTVPG(3) = NODSPC(CNTRL1 + THSMSA)
      LSTVPG(4) = CNTRL1
D     PAUSE ' LEAVING LVSETP 3'
      RETURN
      END
      C
      C
      C
            BLOCK DATA
            IMPLICIT INTEGER(A-Z)
            REAL*4 DEFEXT
            LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
           1      DL2STR,DUMPFL,CURENT,IN2TMP,FD2TMP,DL2TMP,FINDFI,
           2      DLETFI,NSRTFL,REORG,FULL,RPLACE
            COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
           1            INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
           2            LNKSUF,SNKSUF,INSTYP
            COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
           1            FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13.FLAG14,
           2            FLAG15
            COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
           1            MSA,PAGLOC,CURENT
            COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
           1            INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
            COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
           1            USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
            COMMON /LVSWIT/ SETUP,SNGLBK.NXTRAN,IN1STR.IN2STR,FD1STR,FD2STR,
           1            DL1STR,DL2STR.IN2TMP,FD2TMP,DL2TMP,DUMPFI,
           2            FINDFL,DLETFL,NSRTFL
            COMMON /LVPRAM/ BUFLOC.LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
           1            LUN
            COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK.MASKSF.MASKPF
            COMMON /LVUTIL/ FILSPC(39),DEFEXT(2)
            COMMON /LVINS1/ REORG,FULL,RPLACE
      C
            DATA FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,FLAG8,
           1  FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,FLAG15
           2  /"200,"100,"40,"20,"10,"4,"3,
           3    "400,"1000,"2000,"4000,"10000,"20000,"40000,"100000/
            DATA THSMSA,REGAS,PAGENO.CONTNO,INSDEL,USECT,
           1  HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
           2  /1,2,1,2.1,2,1,2,1,2.3,4/
            DATA MWRITE,NOTUSD,NEWCON.FLGMSK,MASKSF,MASKPF
           1  /"4,"2,"1,"7."1777,"176000/
            DATA SUFSZE,NTFREE,FREE,BINARY,BCD,INCLUD,INSTYP,MSADIR.
           1  IPOS,ITYP,NVAL,ITESTR,INDXON,HDRSZE,PAGLOC,OLCHAN
           2  /10,0,1,0,1,0.0,2,
           3    1,3,1,-1,0,2.-1,-1/
            DATA ITYP1/10*0/
            DATA IVALS/10*0/
            DATA SETUP,SNGLBK,NXTRAN,IN1STR.IN2STR,FD1STR,FD2STR,
           1 DL1STR.DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFI,
           2 FINDFI,DLETFI,NSRTFI,REORG.FULL
           3  /.FALSE.,.FALSE.,.FALSE.,.FALSE.,.FALSE...FALSE.,.FALSE.,
           4    .FALSE.,.FALSE.,.FALSE...FALSE.,.FALSE...FALSE.,
           5    .TRUE.,.TRUE...TRUE.,.FALSE.,.FALSE./
            DATA DEFEXT /6RGRFGRF,6RGRFGRF/
      C
      C IF THE FOLLOWING FLAGS ARE ON. THEY REPRESENT THE FOLLOWING:
      C
      C FL0MSK- HEAD OF A MULTIVALUED LIST
      C FL1MSK- THE CELL IS IN WORKING SPACE, NOT AVAILABLE SPACE
      C FL2MSK- VALUE ON A MULTIVALUE LIST
      C FL3MSK- A NODE HAS BEEN DEFINED WITH THIS RELATIVE ADDRESS AS ITS VALUE
      C FL4MSK- THE SAVED INDEX OPERATION IS NOT IN EFFECT FOR THIS LIST
      C FL5MSK- HEAD OF A CONFLICT LIST
      C FLG67 - 00- A RANDOM NUMBER
      C     01- NUMERIC DATA (INTEGER)
      C     10- A CONTINUING STRING OF HOLLERITH DATA
      C     11- THE ONLY, OR FINAL, CELL IN A HOLLERITH DATA STRING
      C FLAG8 - THE CELL CONTAINS A POINTER TO SEQUENCE SPACE
      C FLAG9 - UNUSED
      C FLAG10- MULTIVALUE LIST CONTINUATION FLAG (FUNCTION CONTINUES ON
      C        PREVIOUS CONTINUANT.  THIS CONTINUANT DOES NOT CONTAIN
      C        THE BEGINNING OF THE LIST
      C FLAG11- MULTIVALUE LIST CONTINUATION FLAG (FUNCTION CONTINUES ON
      C     NEXT CONTINUANT)
      C FLAG12- REORG INHIBIT FLAG
      C FLAG13- THE CELL IS THE HEAD OF A MULTIVALUE LIST WHICH IS A
      C     NON-MOVABLE CONTINUATION OF A LIST ON SOME OTHER CONTINUANT
      C
0023        END
```

```
C
C
C
      SUBROUTINE LVFECB
      IMPLICIT INTEGER(A-Z)
      REAL*4 CORE,TOP,BOTTOM
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1      DL2STR,DUMPFL,CURENT,IN2TMP,FD2TMP,DL2TMP,FINDFL,
     2      DLETFL,NSRTFL
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1              INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2              LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1              FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2              FLAG15
      COMMON /LVRAND/ PRIME,SEED,NROW,DNODE,DROW,OLDNOD,LISTSZ,
     1              GRNTBL(256)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2              FINDFL,DLETFL,NSRTFL
      COMMON /LVVSEQ/ ISEQSZ,ISOPOS,LASTSQ,SEQSPC(1)
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1              LUN
      COMMON /LVSTAK/ CURLEV,NUMVAR,STACK(1)
      COMMON /LVRUN/        RUNTYP,CORE
      COMMON /LVUSER/ USER(228)
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE READS A PREVIOUSLY CREATED GRAPH FROM DISK INTO THE GIRS
C BUFFER AND COPIES IT ONTO A NEW DISK FILE.
C
C READ IN SYSTEM VARIABLES
D     PAUSE 'IN LVFECB'
      MSA = 0
      LENGTH = 256
      ERRNUM = 29
      IERR = IREADW(LENGTH,RWBUF(1),MSA,OLCHAN)
      DUMP = 1
      IF(IERR.LT.0) CALL LVERR(DUMP)
C
      REGASP        = RWBUF( 1)
      NXTMSA        = RWBUF( 2)
      PAGSZE        = RWBUF( 3)
      PAGHDR        = RWBUF( 4)
      BUFSZE         = RWBUF( 5)
      DIRSZE         = RWBUF( 6)
      DREGSP         = RWBUF( 7)
      INCORE         = RWBUF( 8)
      HDRSZE         = RWBUF( 9)
      HREQPG         = RWBUF(10)
      HACTPG(1) = RWBUF(11)
      HACTPG(2) = RWBUF(12)
      READCT         = RWBUF(13)
      BLKSZE         = RWBUF(14)
      SUFSZE         = RWBUF(15)
      DIRBLK         = RWBUF(16)
      PRIME          = RWBUF(17)
      SEED           = RWBUF(18)
      LISTSZ         = RWBUF(19)
      ISEQSZ         = RWBUF(20)
        DO 5 I = 1, 4
        CURPAG(I) = RWBUF(20+I)
        LSTVPG(I) = RWBUF(24+I)
5       CONTINUE
C
C READ IN SAVED USER VARIABLES
        DO 7 I = 1,228
        J = I + 28
7       USER(I) = RWBUF(J)
C
      USECNT = 1
      LEASTV = NTFREE
      TOP = INCORE-1
      BOTTOM = INCORE
      CORE = TOP/BOTTOM
C
C READ IN GRN TABLE
      LENGTH = 256
      MSA = 1
      ERRNUM = 30
      IERR = IREADW(LENGTH,GRNTBL(1),MSA,OLCHAN)
      DUMP = 1
      IF(IERR .LT. 0) CALL LVERR(DUMP)
C
C READ IN IN-CORE DIRECTORY
      MSA = 2
      LENGTH = DIRSZE
      ERRNUM = 31
      BUFLOC = 1
      CHAN = OLCHAN
      CALL LVPAGR(CHAN)
C
C COPY OUT-CORE DIRECTORY TO NEW DISK FILE
      DIRPAG = 17
10     DIRPAG = DIRPAG - 1
       CALL LVDRRD(OLCHAN)
       CALL LVDRWR
       IF(DIRPAG .GT. 1) GO TO 10
C
C COPY OLD GRAPH TO NEW DISK FILE
      BUFLOC = DIRSZE + 1
```

```
            ERRNUM = 32
            HIPAGE = HACTPG(1)
            IF(HREQPG .GT. HIPAGE) HIPAGE = HREQPG
      C SEQUENCE ON PAGES
              DO 30 PAGE = 1, HIPAGE
      C SEQUENCE ON CONTINUANTS
            DO 20 I = 1, 64
            CONT = I-1
            REQPAG(1) = PAGE
            REQPAG(2) = CONT
            CALL LVMSA(CONNUM)
            IF(MSARET .LE. 0) GO TO 30
            MSA = MSARET
            CHAN = OLCHAN
            LENGTH = PAGHDR
            CALL LVPAGR(CHAN)
            CALL LVPAGW
      20    CONTINUE
      30      CONTINUE
      C
      C EXAMINE IN-CORE DIRECTORY AND BRING IN LISTED CONTINUANTS INTO CORE
              DO 50 I = 1,DIRSZE
              IF((FLGSPC(I).AND.FL1MSK).EQ.0) GO TO 50
              REQPAG(1) = NODSPC(I)
              REQPAG(2) = I - REQPAG(1) - 1
              IF(REQPAG(2) .LT. 0) REQPAG(2) = REQPAG(2) + DIRSZE
              CTRLPT = LSTSPC(I)
              CTRL1  = CTRLPT + HDRSZE
              CALL LVMSA(CONNUM)
              MSA = MSARET
              BUFLOC = CTRLPT + 1
              LENGTH = PAGHDR
              CALL LVPAGR(CHAN)
      50      CONTINUE
      C
      C ESTABLISH REGISTERS
            CTRLPT = CURPAG(4)
            CTRL1  = CTRLPT + HDRSZE
            CURENT = .TRUE.
            REQPAG(2) = -2
      C READ IN SEQUENCE SPACE (LATER VERSION)
            TYPE 60
      60    FORMAT(/,' GRAPH HAS BEEN PLACED INTO MEMORY',/)
            RETURN
            END
```

```
C
C
C
      SUBROUTINE LVGRN(NODE)
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1      DL2STR,DUMPFL,CURENT,IN2TMP,FD2TMP,DL2TMP,FINDFL,
     2      DLETFL,NSRTFL
      DIMENSION NODE(1)
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1            HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2            DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1            FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2            FLAG15
      COMMON /LVRAND/ PRIME,SEED,NROW,DNODE,DROW,OLDNOD,LISTSZ,
     1            GRNTBL(256)
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1            INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1            USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1            DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2            FINDFL,DLETFL,NSRTFL
      COMMON /LVVTR1/ NODSPC(1)
     1   /LVVTR2/ LSTSPC(1)
     2   /LVVTR3/ LNKSPC(1)
     3   /LVVTR4/ FLGSPC(1)
C
C THE PURPOSE OF THIS ROUTINE IS TO PROVIDE A SEQUENCE OF 'RANDOM'
C NUMBERS OF LENGTH LISTSZ TO THE REQUESTED PAGE (CONTINUANT = 0)
C
C PROGRAM INITIALLIZATION ?
D     PAUSE 'IN LVGRN'
      PAGE = 0
      IF(SETUP) GO TO 100
C
C IS THE PAGE DEFINED ?
      IF(REQPAG(1) .GT. HACTPG(1)) HACTPG(1) = REQPAG(1)
      IF(REQPAG(1) .GT. 0) GO TO 50
      HACTPG(1) = HACTPG(1) + 1
      REQPAG(1) = HACTPG(1)
C
C OBTAIN CURRENT VALUES FOR GRN PARAMETERS FOR REQUESTED PAGE
50    PAGE = REQPAG(1)
      GRNDEX = 4*(PAGE - 1)
      OLDNOD = GRNTBL(GRNDEX + OLDNDH)
      DNODE  = GRNTBL(GRNDEX + DNODEH)
      NROW   = GRNTBL(GRNDEX + NROWH)
      DROW   = GRNTBL(GRNDEX + DROWH)
C
100     DO 200 J = 1,LISTSZ
        I = J
        NODE(I) = OLDNOD+DNODE
        OLDNOD = NODE(I)
        DNODE = DNODE+1
        IF(NODE(I).LE.PAGSZE) GO TO 199
```

```
C RESIDUE GENERATION ?
            IF(NROW.GT.PRIME) GO TO 150
C ROW UPDATE
            NROW = NROW+SEED
            IF(NROW.GT.PRIME) NROW = NROW-PRIME
            NODE(I) = NROW
            OLDNOD = NODE(I)
            DNODE = PRIME+1
C RESIDUE GENERATION ?
            IF(NODE(I).NE.SEED) GO TO 199
            NROW = 0
            DROW = PRIME
C RESIDUE GENERATION
150         DROW = DROW+1
            NROW = NROW +DROW
            NODE(I) = NROW
            OLDNOD = NODE(I)
            DNODE = DROW
            IF(NODE(I).GT.PAGSZE) GO TO 300
C OUTPUT NODE
199         IF(SETUP .EQ. .TRUE.) RETURN
200         NODE(I) = NODE(I).OR.LVLFSH(PAGE,SUPSZE)
C
C UPDATE HEADER
250      GRNTBL(GRNDEX + OLDNDH) = OLDNOD
         GRNTBL(GRNDEX + DNODEH) = DNODE
         GRNTBL(GRNDEX + NROWH)  = NROW
         GRNTBL(GRNDEX + DROWH)  = DROW
         LISTSZ=1
         RETURN
C
C ORIGINAL CREATION SEQUENCE IS EXHAUSTED, RECOVER UNDEFINED NODES
C   BRING IN CONTINUANT ZERO OF THE REQUESTED PAGE IF NECESSARY
300      REQCON = REQPAG(2)
         REQPAG(2) = 0
         CALL LVEXCH
           DO 430 L = 1,LISTSZ
         DO 400 K=1,PAGSZE
         LOC = CTRIPT + HDRSZE + K
         IF((FLGSPC(LOC).AND.FL3MSK).NE.0) GO TO 390
         NODE(L)=K
         FLGSPC(LOC)=FLGSPC(LOC).AND.FL3MSK
         GO TO 430
390      IF(K .EQ. PAGSZE) GO TO 440
400      CONTINUE
430        CONTINUE
C BRING IN ORIGINAL REQUESTED (PAGE,CONT)
         REQPAG(2) = REQCON
         CALL LVEXCH
         LISTSZ = 1
         RETURN
C
440      TYPE 450
450      FORMAT(1H ,'ERROR...NUMBER OF NODES EXCEEDS REQUESTED MEMORY,'/'
        1 PROGRAM IS TERMINATED.')
         ERRNUM = 10
         DUMP = 0
         CALL LVERR(DUMP)
         STOP
         END
```

```
C
C
C
      SUBROUTINE LVNPAG
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1       DL2STR,DUMPFI,CURENT,IN2TMP,FD2TMP,DL2TMP,FINDFL,
     2       DLETFI,NSRTFI
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,SUPSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFI,
     2              FINDFI,DLETFI,NSRTFI
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1              LUN
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE WILL PLACE A NEW PAGE (ZEROTH CONTINUANT) INTO THE BUFFER.
C IF THE NEW PAGE EXCEEDS THE NUMBER OF PAGES ORIGINALLY REQUESTED BY THE
C USER:  PUT IT ON DISK.
C REGISTERS AND IN-CORE AND OUT-CORE DIRECTORIES ARE UPDATED.
C
C DEFINE PAGE NO. AND UPDATE HIGHEST ACTIVE PAGE
D     PAUSE 'IN LVNPAG'
      HACTPG(1) = HACTPG(1)+1
      REQPAG(1) = HACTPG(1)
C
C BRING IN OUT-CORE DIRECTORY PAGE AND DEFINE OUTLOC (LOC IN O-C D P)
      REQPAG(2) = -1
      CALL LVMSA(CONNUM)
      REQPAG(2) = 0
C
C ARE ANY PREALLOCATED PAGES (THAT HAVE ALREADY BEEN OPENED ON DISK)
C STILL AVAILABLE ?
      IF(HACTPG(1) .GT. HREQPG) GO TO 10
C
C REQ(P,0) WAS CREATED AT THE BEGINNING OF THE PROGRAM
      CALL LVEXCH
      HACTPG(2) = MSARET
      RETURN
C
C OPEN A PAGE-BLOCK IN THE BUFFER
10    IF(LEASTV .EQ. NTFREE) CALL LVOPEN
C
C*** NEW PAGE MUST BE ADDED TO THE DISK IMMEDIATELY FOLLOWING THE LAST
C CREATED CONTINUANT.
```

```
      HACTPG(2) = NXTMSA
C
C PLACE LOCATION OF NEW PAGE INTO OUT-CORE DIRECTORY
      OUTDIR(OUTLOC) = HACTPG(2)
C
C SET UP AVAILABLE SPACE AND HEADER
      SNGLBK = .TRUE.
      CTRLPT = LSTVPG(4)
      CTRL1  = CTRLPT + HDRSZE
      CALL LVSETP
C
C PLACE EMPTY PAGE ON DISK
      LENGTH = PAGHDR
      BUFLOC = CTRLPT + 1
      ERRNUM = 23
      MSA = HACTPG(2)
      CALL LVPAGW
C
C UPDATE REGISTERS
 40   CURPAG(1) = HACTPG(1)
      CURPAG(2) = 0
      CURPAG(3) = HACTPG(2)
      CURPAG(4) = LSTVPG(4)
C
C PAGE HAS BEEN PLACED IN "LEAST VALUED BLOCK"
C   UPDATE IN-CORE DIRECTORY
      CALL LVRPLC
C
C PROTECT THIS PAGE FROM BEING TAKEN OUT OF CORE BEFORE IT IS USED
      FLGSPC(CTRLPT+HDRFLG) = NOTUSD .OR. NEWCON
      RETURN
      END
      C
      C
      C
0001        SUBROUTINE LVNCON
0002        IMPLICIT INTEGER(A-Z)
0003        LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR.
           1      DL2STR,DUMPFL,CURENT,IN2TMP,FD2TMP,DL2TMP,FINDFL,
           2      DLETFL,NSRTFL
0004        COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
           1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
           2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
0005        COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
0006        COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
           1             MSA,PAGLOC.CURENT
0007        COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
           1             INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
0008        COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
           1             USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
0009        COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
           1             DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
           2             FINDFL,DLETFL,NSRTFL
0010        COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
           1             LUN
0011        COMMON /LVVTR1/ NODSPC(1)
           1    /LVVTR2/ LSTSPC(1)
           2    /LVVTR3/ LNKSPC(1)
           3    /LVVTR4/ FLGSPC(1)
0012        DIMENSION CONLST(64)
0013        DATA CONLST /64*0/
      C
      C THIS ROUTINE PLACES AN UNUSED CONTINUANT OF AN ESTABLISHED PAGE
      C INTO THE BUFFER.  IF REQ(P,C) WAS NOT INITIALIZED AT THE BEGINNING
      C OF THE PROGRAM, A CONTINUANT IS CREATED AND PLACED ON DISK.
      C REGISTERS, IN-CORE, AND OUT-CORE DIRECTORIES ARE UPDATED.
      C
      C CONLST() IS A LIST OF HIGHEST ACTIVE CONTINUANTS FOR EACH PAGE
      D     PAUSE 'IN LVNCON'
0014        PAGE = REQPAG(1)
0015        HICONT = CONLST(PAGE)
0016        REQPAG(2) = HICONT + 1
0017        CONLST(PAGE) = REQPAG(2)
      C
      C BRING IN OUTCORE DIRECTORY
0018        CALL LVMSA(CONNUM)
      C
      C OPEN A PAGE-BLOCK IN THE BUFFER
0019        IF(LEASTV .EQ. NTFREE) CALL LVOPEN
      C
      C ARE ANY PREINITIALIZED CONTINUANTS STILL AVAILABLE ?
0021        IF(MSARET .GT. 0) GO TO 10
      C
      C*** NEW CONTINUANT MUST BE ADDED TO THE DISK IMMEDIATELY FOLLOWING THE
      C LAST CREATED CONTINUANT.
0023        MSA = NXTMSA
      C
      C PLACE LOCATION OF NEW PAGE INTO OUT-CORE DIRECTORY
0024        OUTDIR(OUTLOC + 1) = MSA
```

```
C
C SET UP AVAILABLE SPACE AND HEADER
      SNGLBK = .TRUE.
      CTRLPT = LSTVPG(4)
      CTRL1  = CTRLPT + HDRSZE
      CALL LVSETP
C
C PLACE EMPTY PAGE ON DISK
      LENGTH = PAGHDR
      BUFLOC = CTRLPT + 1
      ERRNUM = 25
      CALL LVPAGW
      GO TO 40
C
C REQ(P,0) WAS CREATED AT THE BEGINNING OF THE PROGRAM
C READ REQ(P,0) INTO CORE
 10   MSA = MSARET
      LENGTH = PAGHDR
      BUFLOC = LSTVPG(4) + 1
      ERRNUM = 26
      CHAN = NWCHAN
      CALL LVPAGR(CHAN)
C
C UPDATE REGISTERS
 40   CURPAG(1) = REQPAG(1)
      CURPAG(2) = REQPAG(2)
      CURPAG(3) = MSA
      CURPAG(4) = LSTVPG(4)
C
C PAGE HAS BEEN PLACED IN "LEAST VALUED BLOCK"
C   UPDATE IN-CORE DIRECTORY
      CALL LVRPLC
C
C PROTECT THIS PAGE FROM BEING TAKEN OUT OF CORE BEFORE IT IS USED
      FLGSPC(CTRLPT+HDRFLG) = NOTUSD .OR. NEWCON
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVEXCH
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1              INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2              LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1              LUN
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVVTR1/ NODSPC(1)
     1   /LVVTR2/ LSTSPC(1)
     2   /LVVTR3/ LNKSPC(1)
     3   /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE BRINGS THE REQUESTED (PAGE,CONT) INTO CORE IF NECESSARY
C AND UPDATES THE IN-CORE DIRECTORY AND "CURRENT" REGISTER
C TO REQUESTED (P,C)
C FAILURE RETURN IF MSARET LT 0
C
C IS REQ(P,C) IN CORE ?
D     PAUSE 'IN LVEXCH'
D     TYPE 1
D1    FORMAT(' REQ(1) REQ(2)')
D     TYPE 2,REQPAG(1),REQPAG(2)
D2    FORMAT(2(2X,I4))
      MSARET = 10000
      CALL LVDRCT
      IF(PAGLOC .GT. 0) RETURN
C
C BRING REQ(P,C) INTO CORE IF IT EXISTS
C   LOCATE REQ(P,C) ON DISK
      CALL LVMSA(CONT)
C   DOES REQ(P,C) EXIST ?
      IF(MSARET .LT. 0) GO TO 100
C   MAKE A PAGE-BLOCK AVAILABLE IN THE GIRS BUFFER
      CALL LVOPEN
C   BRING REQ(P,C) INTO "LEAST-VALUED" PAGE-BLOCK
      ERRNUM = 10
      LENGTH = PAGHDR
      MSA = MSARET
      BUFLOC = LSTVPG(4) + 1
      CHAN = NWCHAN
      CALL LVPAGR(CHAN)
C UPDATE "CURRENT PAGE" REGISTERS
      CTRLPT = BUFLOC - 1
      CTRL1  = CTRLPT + HDRSZE
      PAGLOC = CTRLPT
      CURPAG(1) = LSTSPC(CTRLPT + PAGENO)
      CURPAG(2) = LSTSPC(CTRLPT + CONTNO)
      CURPAG(3) = MSA
      CURPAG(4) = CTRLPT
      CURENT = .TRUE.
      REGASP = NODSPC(CTRLPT + REGAS)
C UPDATE IN-CORE DIRECTORY
      CALL LVRPLC
C UPDATE (P,C) HEADER
      READCT = READCT + 1
      FLGSPC(CTRLPT+READVL) = READCT
      LNKSPC(CTRLPT+USECT) = 0
 100  LEASTV = NTFREE
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVDRCT
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1                INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2                LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1                HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2                DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1                MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1                INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1                USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1                LUN
      COMMON /LVSTAK/ CURLEV,NUMVAR,STACK(1)
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
      DIMENSION TEMP(4)
C
C THIS ROUTINE SEARCHES THE DIRECTORY TO SEE IF THE REQUESTED PAGE-
C CONTINUANT (REQPAG) IS IN CORE.  THE DIRECTORY IS A PAGE-BLOCK WHICH
C STAYS IN CORE.  IT IS AT THE FIRST CONTROL POINT (CTRLPT = 0).  DIRSZE
C IS THE DIRECTORY PAGE SIZE.  FOR EACH PAGE-CONTINUANT THAT IS IN
C CORE, A TRIPLE IS STORED.  THE SOURCE NODE IS THE CONTINUANT+1, THE
C LINK (OR KEY) IS THE PAGE NUMBER, AND THE VALUE IS THE LOCATION
C PRECEEDING THE FIRST WORD OF THAT PAGE IN CORE ( = CTRLPT).
C
C  SUCCESS -- UPDATE CURPAG AND CTRLPT, CURENT = .TRUE.
C  FAILURE -- PAGLOC = -1
C
C*** DOES THE REQUESTED PAGE-CONTINUANT = THE CURRENT PAGE-CONTINUANT ?
C
D     PAUSE 'IN LVDRCT'
      IF((REQPAG(1) .EQ. CURPAG(1)) .AND. (REQPAG(2) .EQ. CURPAG(2)))
     1 GO TO 40
      IF(INCORE .EQ. 1) GO TO 98
      GO TO 50
C
 40   CTRLPT = CURPAG(4)
      CTRL1 = CTRLPT + HDRSZE
      PAGLOC = CTRLPT
      MSA = CURPAG(3)
      REGASP = NODSPC(CTRLPT + REGAS)
      CURENT = .TRUE.
      RETURN
C
C
C*** TEMPORARILY STORE SYSTEM VARIABLES FOR THE SEARCH
C
```

```
50    CALL LVSTAC
      OLDCPT = CTRLPT
C
C RESET SYSTEM VARIABLES FOR THE DIRECTORY
C
      CTRLPT = -HDRSZE
      CTRL1  = 0
      PAGSZE = DIRSZE
      IARG   = REQPAG(2)+1
      IFUNC  = REQPAG(1)
      SRCSUF = IARG
      LNKSUF = IFUNC
C
      CALL LVFIND
      PAGLOC = ITESTR*IVAL
C
C RESTORE SYSTEM VARIABLES
      CALL LVPOP
      IF(PAGLOC .LT. 0) GO TO 99
C
C PAGE-CONTINUANT HAS BEEN FOUND IN THE DIRECTORY.
      CTRLPT = PAGLOC
      CTRL1 = CTRLPT + HDRSZE
      MSA = NOUSPC(CTRLPT + THSMSA)
C
C UPDATE CURPAG
      CURPAG(1) = REQPAG(1)
      CURPAG(2) = REQPAG(2)
      CURPAG(3) = MSA
      CURPAG(4) = CTRLPT
      CURENT = .TRUE.
      REGASP = NOUSPC(CTRLPT + REGAS)
      RETURN
C
C FAILURE
98    PAGLOC = -1
99    CURENT = .FALSE.
      CTRLPT = OLDCPT
      CTRL1  = CTRLPT + HDRSZE
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVMSA(CONNUM)
      IMPLICIT INTEGER(A-Z)
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1            HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2            DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1            INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
C
C THIS ROUTINE BRINGS INTO OUTDIR() THE CORRECT OUTCORE DIRECTORY BLOCK
C IF NECESSARY.  UPDATES DIRPAG AND DIRCNT.
C
C SUCCESS:
C     RETURNS MSA OF [REQPAG(1),REQPAG(2)] IN MSARET.
C     SETS CONNUM = REQPAG(2).
C FAILURE OR NEW CONTINUANT:
C     MSARET = -1
C     CONNUM = HIGHEST EXISTING CONTINUANT NUMBER OF REQPAG(1)
C    UNDEFINED PAGE:
C     CONNUM = -1
C
D     PAUSE 'IN LVMSA'
      PAGE = REQPAG(1)
      CONT = REQPAG(2)
      ERRNUM = 11
      DUMP = 0
      IF((PAGE .GT. 64) .OR. (CONT .GT. 63)) CALL LVERP(DUMP)
C COMPUTE OUTCORE DIRECTORY BLOCK
      NEWDIR = (PAGE - 1)/4 + 1
      DIRCNT = PAGE - 4*(NEWDIR - 1)
C BRING IN DIRECTORY BLOCK IF NECESSARY
      IF(NEWDIR .EQ. DIRPAG) GO TO 100
      DIRPAG = NEWDIR
      CHAN = NWCHAN
      CALL LVDRRD(CHAN)
C
C DETERMINE IF "ANY", "SPECIFIC", OR "NEW" CONTINUANT IS REQUESTED
100   IF(CONT + 1) 200, 300, 210
C
C "ANY" -- SET TO ZERO
200   CONT = 0
C
C "SPECIFIC"
210   OUTLOC = 1 + 64*(DIRCNT-1) + CONT
      MSARET = OUTDIR(OUTLOC)
D     TYPE 9
D9    FORMAT(' REQPAG(1),REQPAG(2),DIRPAG,DIRCNT,CONT,OUTLOC,MSARET')
D     TYPE 10, REQPAG(1),REQPAG(2),DIRPAG,DIRCNT,CONT,OUTLOC,MSARET
D10   FORMAT(1X,8(2X,I5))
      CONNUM = CONT
      IF(CONT .LT. 0) GO TO 220
      IF(MSARET .GT. 0) GO TO 220
      CONT = CONT - 1
      GO TO 210
C FAILURE ?
```

```
220   IF(CONT .NE. REQPAG(2)) MSARET = -1
      RETURN
C
C "NEW CONTINUANT"  (PAGE MUST BE DEFINED, CONT NOT YET INITIALIZED)
300   MSARET = -1
      CONNUM = -1
310   CONT = CONT + 1
      IF(CONT .GE. 64) STOP 'REQUEST EXCEEDS ALLOWABLE NUMBER OF
     1 CONTINUANTS'
      OUTLOC = 1 + 64*(DIRCNT-1) + CONT
      MSARET = OUTDIR(OUTLOC)
      IF(MSARET .LE. 0) RETURN
      CONNUM = CONT
      GO TO 310
      END
```

```
C
C
C
      SUBROUTINE LVOPEN
      IMPLICIT INTEGER(A-Z)
      REAL*4 CORE
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1             MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1             INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1             USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1             LUN
      COMMON /LVRUN/       RUNTYP,CORE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
C THE PURPOSE OF THIS ROUTINE IS TO MAKE A PAGE-BLOCK AVAILABLE
C IN WRKSPC.
C
C IF LEASTV = FREE, THEN LSTVPG CONTAINS THE CONTROL POINT FOR AN
C AVAILABLE PAGE-BLOCK
C
D     PAUSE 'IN LVOPEN'
      IF(LEASTV .EQ. FREE) RETURN
C
C PREPARE TOTAL USAGE COUNT FOR LVALUE
      CALL LVSUM
C
C ROUTINE LVALUE WILL RETURN THE DISK AND IN-CORE LOCATIONS IN LSTVPG()
C OF A PAGE, CONT WHICH IT HAS DETERMINED TO BE OF LEAST IMMEDIATE USE
C TO THE SYSTEM.
C
      CALL LVALUE
C
C TEST WRITE-BIT OF LEAST VALUED CONTINUANT. THE WRITE-BIT INDICATES
C WHETHER OR NOT A PAGE HAS BEEN CHANGED SINCE IT WAS READ INTO
C MEMORY
C
      FLGLOC = LSTVPG(4) + HDRFLG
      IF((MWRITE .AND. FLGSPC(FLGLOC)) .EQ. 0) RETURN
C
C WRITE LEAST VALUED PAGE TO DISK
C
C   ZERO OUT FLAGS
      FLGSPC(FLGLOC) = 0
      MSA = LSTVPG(3)
      LENGTH = PAGHDR
      ERRNUM = 27
      BUFLOC = LSTVPG(4) + 1
      CALL LVPAGW
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVSUM
      IMPLICIT INTEGER(A-Z)
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR.BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,SUPSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE IS CALLED WHENEVER A CONTINUANT IS CREATED OR READ
C INTO MEMORY.  ITS RESULTS ARE USED BY THE LVALUE ROUTINE FOR THE
C 'USAGE' PARAMETER.
D     PAUSE 'IN LVSUM'
      USECNT = 0
      MBIAS = DIRSZE
C RECOMPUTE USE COUNT OF ALL INCORE CONTINUANTS
        DO 10 I = 1,INCORE
        USECNT = USECNT + LNKSPC(MBIAS + USECT)
        MBIAS = MBIAS + PAGHDR
 10     CONTINUE
      IF(USECNT .EQ. 0) USECNT = 1
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVALUE
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT
      REAL*4 A,B,C,D,BOTOM1,BOTTOM,CORE,ORDER,TOP,TOP1,TTLUSE,USAGE,
     1      USAGE1,VALUE,VALUE1,WRITE,SPACE,CAPACY
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR.FD2STR,
     1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2              FINDFL,DLETFL,NSRTFL
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK.MASKSF,MASKPF
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1              LUN
      COMMON /LVRUN/        RUNTYP,CORE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
      DATA A,B,C,D/15.0,20.0,15.0,50.0/
C
C THIS ROUTINE WILL DETERMINE WHICH CONTINUANT IS LEAST NEEDED IN
C CORE.  THE ALGORITHM USED IS A MODIFICATION OF THE INTERACTIVE DATA
C MANAGER OPTIMIZATION ALGORITHM WRITTEN BY MEL HAAS, CODE 1833 .
C THE VALUES RANGE FROM 0 (LEAST NEEDED CONTINUANT) TO 100 (MOST
C USEFUL CONTINUANT).
D     PAUSE 'IN LVALUE'
      JBIAS = DIRSZE
      MBIAS = JBIAS
      IF(INCORE .EQ. 1) GO TO 20
      LEASTV = NTFREE
      VALUE = 100000.0
        DO 10 I = 1,INCORE
        IF((FLGSPC(JBIAS + HDRFLG) .AND. NOTUSD) .NE. 0) GO TO 9
D     PAUSE ' IN LVALUE LOOP'
C
C CALCULATE ORDER VALUE
        INPOS = FLGSPC(JBIAS + READVL)
        TOP = READCT-INPOS
        BOTTOM = READCT
        ORDER = 1.0 - (TOP/BOTTOM)
C
C CALCULATE WRITE VALUE
        WRITE = 0.0
        IF((FLGSPC(JBIAS + HDRFLG) .AND. MWRITE) .NE. 0) WRITE = CORE
C
C CALCULATE USAGE VALUE
        TTLUSE = USECNT
        USAGE1 = LNKSPC(JBIAS + USECT)
        USAGE = USAGE1/TTLUSE
C
C CALCULATE SPACE VALUE
        TOP1 = PAGSZE - LNKSPC(JBIAS + INSDEL)
        BOTOM1 = PAGSZE
        CAPACY = TOP1/BOTOM1
        GO TO (1,2,3), RUNTYP
C
C CREATION TYPE RUN
1       SPACE = -4.0*CAPACY*(CAPACY-1.0)
        GO TO 5
C
C PRODUCTION TYPE RUN
2       SPACE = 1.0
        IF( CAPACY .LT. .125) SPACE = 8.0*CAPACY
        IF( CAPACY .GE. .750) SPACE = 0.0
        IF((CAPACY .GT. .375).AND.(CAPACY .LT. .75))
     1    SPACE = 2.0-(8.0/3.0)*CAPACY
        GO TO 5
C
C QUERY TYPE RUN
3       SPACE = 1.0-CAPACY
C
C*** CALCULATE THE IN-CORE VALUE OF THIS CONTINUANT
C
5       VALUE1 = A*ORDER+B*USAGE+C*SPACE+D*WRITE
D     TYPE 111,ORDER,USAGE,SPACE,WRITE,VALUE1,VALUE
D111  FORMAT(1X,6(F10.3,2X))
        IF(VALUE1 .GE. VALUE) GO TO 9
        LEASTV = FREE
        VALUE = VALUE1
        MBIAS = JBIAS
9       JBIAS = JBIAS+PAGHDR
10      CONTINUE
C
C UPDATE 'LEAST VALUED' REGISTER LSTVPG UNLESS NONE WAS FOUND
      IF(LEASTV .EQ. FREE) GO TO 20
      TYPE 15
15    FORMAT(/,' *** ERROR IN LVALUE ***',/)
C
C PREVENT POSSIBLE RECURSION
      IF(DUMPFL .EQ. TRUE) STOP
      IERR = 0
      ERRNUM = 28
      MODE = BCD
      PAGES = -1
      DUMP = 0
      CALL LVDUMP(DUMP)
      STOP
C
20    LSTVPG(1) = LSTSPC(MBIAS + PAGENO)
      LSTVPG(2) = LSTSPC(MBIAS + CONTNO)
      LSTVPG(3) = NODSPC(MBIAS + THSMSA)
      LSTVPG(4) = MBIAS
C SET LEASTV FOR INCORE = 1
      LEASTV = FREE
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVRPLC
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT,REORG,FULL,RPLACE
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1            INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2            LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1            HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2            DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1            MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCBAN,OLCBAN,CWPAND,PAGHDR,BUFSZE,DIRSZE,
     1            INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVDEL1  NUMRET
      COMMON /LVINS1/ REORG,FULL,RPLACE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE UPDATES THE DIRECTORY BY DELETING LSTVPG (THE LEAST
C VALUED BLOCK IN CORE) FROM IT, AND THEN INSERTING CURPAG (THE NEW
C CURRENT PAGE) INTO IT.
C
C*** SAVE SYSTEM VARIABLES
C
D     PAUSE 'IN LVRPLC'
      CALL LVSTAC
C
C*** DELETE OLD PAGE,CONTINUANT, LSTVPG, FROM DIRECTORY
C
      CTRLPT = -HDRSZE
      CTRL1  = 0
      PAGSZE = DIRSZE
      IF(LSTVPG(1) .EQ. 0) GO TO 5
      IARG   = LSTVPG(2) + 1
      IFUNC = LSTVPG(1)
      SRCSUF = IARG
      LNKSUF = IFUNC
      CALL LVFIND
      ITYP = 1
      IPOS = 1
      INDXON = 0
      REORG = .FALSE.
      NUMRET = 0
      TEMP = REGASP
      REGASP = DREGSP
      CALL LVDLET
C
C*** PLACE NEW PAGE-CONTINUANT INTO DIRECTORY
C
    5 IARG   = CURPAG(2) + 1
      IFUNC = CURPAG(1)
      SRCSUF = IARG
      LNKSUF = IFUNC
      CALL LVFIND
      INSTYP = 1
      NVAL = 1
      INDXON = 0
      ITYP1(1) = 1
      IVALS(1) = CURPAG(4)
      SNKSUF = CURPAG(4)
      CALL LVNSRT
C
C*** RESTORE SYSTEM VARIABLES
C
      DREGSP = REGASP
      REGASP = TEMP
      CALL LVPOP
      CTRLPT = CURPAG(4)
      CTRL1  = CTRLPT + HDRSZE
C
C PAGE-BLOCK HAS BEEN FILLED AND IS NO LONGER AVAILABLE
      LEASTV = NTFREE
      RETURN
      END
```

```
C
C
C
          SUBROUTINE LVSTAC
          IMPLICIT INTEGER(A-Z)
          DIMENSION STORE(1),STOR(6)
          COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
         1             INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
         2             LNKSUF,SNKSUF,INSTYP
          COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
         1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
         2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
          COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
         1             INCORE,HDRSZE.MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
          COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
         1             LUN
          COMMON /LVADDR/ IADD,THIS,LSTHED,LOC.LAST,LASTLC
          COMMON /LVSTAK/ CURLEV,NUMVAR,STACK(140)
          EQUIVALENCE (IFUNC,STORE(1))
          EQUIVALENCE (IADD,STOR(1))
          DATA NUMVAR,MAXLEV,CURLEV/34,3,0/
C
C THIS ROUTINE SAVES UP TO 3 SETS OF /LVARGS/ VARIABLES AND REQUEST REGISTERS
D         PAUSE 'IN LVSTAC'
          CURLEV = CURLEV + 1
          IF(CURLEV .GT. MAXLEV) GO TO 99
          ISTLOC = (CURLEV-1)*(NUMVAR + 11) + 1
           DO 10 I = 1,NUMVAR
           STACK(I+ISTLOC) = STORE(I)
 10        CONTINUE
           DO 20 I = 1,4
           STACK(NUMVAR+ISTLOC+I) = REQPAG(I)
 20        CONTINUE
           DO 30 I = 1,6
           STACK(NUMVAR+ISTLOC+I+4) = STOR(I)
 30        CONTINUE
          STACK(NUMVAR+ISTLOC+11) = PAGSZE
          RETURN
C
C FAILURE - ATTEMPT TO STACK TOO MANY SETS OF VARIABLES
 99       ERRNUM = 21
          DUMP = 1
          CALL LVERR(DUMP)
          RETURN
          END
```

```
C
C
C
      SUBROUTINE LVPOP
      IMPLICIT INTEGER(A-Z)
      DIMENSION STORE(1),STOR(6)
      COMMON /IVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1                INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2                LNKSUF,SNKSUF,INSTYP
      COMMON /IVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSAREF,
     1                HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2                DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /IVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1                INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /IVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1                LUN
      COMMON /IVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /IVSTAK/ CURLEV,NUMVAR,STACK(1)
      EQUIVALENCE (IFUNC,STORE(1))
      EQUIVALENCE (IADD,STOR(1))
C
C THIS ROUTINE RETURNS UP TO 3 SETS OF /IVARGS/ VARIABLES
D     PAUSE 'IN LVPOP'
      CURLEV = CURLEV - 1
      IF(CURLEV .LT. 0) GO TO 99
      ISTLOC = (CURLEV)*(NUMVAR+11)+1
       DO 10 I = 1,NUMVAR
       STORE(I) = STACK(I+ISTLOC)
10     CONTINUE
       DO 20 I = 1,4
       REQPAG(I) = STACK(I+ISTLOC+NUMVAR)
20     CONTINUE
       DO 30 I = 1,6
       STOR(I) = STACK(I+ISTLOC+NUMVAR+4)
30     CONTINUE
      PAGSZE = STACK(NUMVAR+ISTLOC+11)
      RETURN
C
C FAILURE - ATTEMPT TO RETURN TOO MANY SETS OF VARIABLES
99    ERRNUM = 22
      DUMP = 1
      CALL LVERR(DUMP)
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVREOR(REQCON)
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1      DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,FD1TMP,
     2      DL2TMP,IN2TMP,FD2TMP,INSIDE,FULL,REORG,LSTCON,RPLACE,
     3      FINISH
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1            INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2            LNKSUF,SNKSUF,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1            HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2            DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1            FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2            FLAG15
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1            MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1            INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDFL,
     1            USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWD
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1            DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2            FINDFL,DLETFL,NSRTFL
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1            LUN
      COMMON /LVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /LVEND/        COUNT,ABSPOS,LSTCON
      COMMON /LVINS1/ REORG,FULL,RPLACE
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
D     PAUSE 'IN LVREOR'
C
C REORG MOVES THE LIST FROM ITS PRESENT LOCATION (CONTINUANT = REQPAG(2))
C TO CONTINUANT REQCON.
      ERRNUM = 62
      DUMP = 0
      IF(REQCON .EQ. CURPAG(2)) CALL LVERR(DUMP)
C
C WAS THE ORIGINAL LIST PLACED ON CURPAG(2) AS A RESULT OF A SPECIFIC
C REQUEST ?   IF SO, ERROR.
      IF((FLGSPC(CTRL1+THIS) .AND. FLAG12) .EQ. 0) GO TO 20
C
C TYPE OUT ERROR, BUT PROCEED ANYWAY
      TYPE 10, SRCSUF,LNKSUF,IVALS(1),REQPAG(1),REQCON,CURPAG(2)
10    FORMAT(/,' *** ERROR ***, THE FOLLOWING TRIPLE ',3(O6,2X),
     1 ' OF PAGE ',I3,' WILL BE PLACED ON CONTINUANT ',I3,/,
     2 ' ORIGINAL LIST WAS FOUND ON CONTINUANT ',I3)
C
20    OLDCON = CURPAG(2)
```

```
      OLDCPT = CTRLPT
      OLDCP1 = CTRL1
C
C SPECIAL HANDLING IF BUFFER HOLDS ONLY ONE CONTINUANT
      IF(INCORE .EQ. 1) GO TO 30
C
C KEEP CONTINUANT WITH OLD LIST IN CORE
      FLGSPC(CTRLPT+HDRFLG) = FLGSPC(CTRLPT+HDRFLG) .OR. NOTUSD
C
C LOCATE NEW CONTINUANT
      REQPAG(2) = REQCON
      CALL LVEXCH
30    NEWCPT = CTRLPT
      NEWCP1 = CTRL1
C
C SAVE THE VALUE TO BE INSERTED
      CALL LVSTAC
C
      IF(LSTHED .GT. 0) GO TO 50
C SVI
      KVAL = 1
      IVALS(1) = LSTSPC(OLDCP1+THIS)
      ITYP1(1) = FLGSPC(OLDCP1+THIS) .AND. FLG67
      IF(INCORE .GT. 1) GO TO 45
      REQPAG(2) = REQCON
      CALL LVEXCH
45    CALL LVFIND
      CALL LVNSRT
      NODSPC(CTRLPT+REGAS) = REGASP
      FLGSPC(CTRLPT+HDRFLG)=FLGSPC(CTRLPT+HDRFLG) .OR. MWRITE
      GO TO 70
C MVI
50    OLDLOC = LSTHED
      KVAL = 0
      MVAL = 0
      IF(INCORE .EQ. 1) GO TO 63
60    KVAL = KVAL + 1
      IVALS(1) = NODSPC(OLDCP1+OLDLOC)
      ITYP1(1) = FLGSPC(OLDCP1+OLDLOC) .AND. FLG67
      CALL LVFIND
      CALL LVNSRT
      OLDLOC = LSTSPC(OLDCP1+OLDLOC)
      IF((FLGSPC(OLDCP1+OLDLOC) .AND. FL0MSK) .EQ. 0) GO TO 60
      GO TO 70
C
C FOR MVI (INCORE = 1) SWAP OLD AND NEW CONTINUANTS IN AND OUT WHILE
C RE-INSERTING UP TO TEN VALUES AT A TIME
63    FINISH = .FALSE.
65    REQPAG(2) = OLDCON
      CALL LVEXCH
66    MVAL = MVAL + 1
      KVAL = KVAL + 1
      IVALS(KVAL) = NODSPC(OLDCP1+OLDLOC)
      ITYP1(KVAL) = FLGSPC(OLDCP1+OLDLOC) .AND. FLG67
      OLDLOC = LSTSPC(OLDCP1+OLDLOC)
      IF((FLGSPC(OLDCP1+OLDLOC) .AND. FL0MSK) .NE. 0) FINISH = .TRUE.
      IF(KVAL .GE. 10) GO TO 67
```

```
            IF(.NOT. FINISH) GO TO 66
      67    REQPAG(2) = REQCON
            CALL LVEXCH
            CALL LVFIND
            NVAL = KVAL
            KVAL = 0
            CALL LVNSRT
            NODSPC(CTRLPT+REGAS) = REGASP
            FLGSPC(CTRLPT+HDRFLG)=FLGSPC(CTRLPT+HDRFLG) .OR. MWRITE
            IF(.NOT. FINISH) GO TO 65
            KVAL = MVAL
            REQPAG(2) = OLDCON
            CALL LVEXCH
      C
      C DELETE OLD LIST
      70    INDXON = 0
            CTRLPT = OLDCPT
            CTRL1  = OLDCP1
            IF(INCORE .GT. 1) GO TO 75
            REQPAG(2) = OLDCON
            CALL LVEXCH
      75    CALL LVFIND
            CALL LVDLET
      C
      C UPDATE HEADER
      C OLD CONTINUANT HAS BEEN MODIFIED
            FLGSPC(CTRLPT+HDRFLG)=FLGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSD
           1 .OR. MWRITE
      C
      C RESET CONTINUANT USAGE RATIO
            LNKSPC(OLDCPT+INSDEL) = LNKSPC(OLDCPT+INSDEL) - KVAL
      C
      C INSERT NEW VALUE
            CALL LVPOP
            REQPAG(2) = REQCON
            IF(INCORE .EQ. 1) CALL LVEXCH
            CTRLPT = NEWCPT
            CTRL1  = NEWCP1
            CALL LVFIND
            CALL LVNSRT
      C
      C UPDATE HEADER
      C RESET CONTINUANT USAGE RATIO
            LNKSPC(NEWCPT+INSDEL) = LNKSPC(NEWCPT+INSDEL) + KVAL
            RETURN
            END
```

```
C
C
C
      SUBROUTINE LVINCL
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1         DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,
     2         DL2TMP,IN2TMP,FD2TMP,REORG,FULL,RPLACE
      COMMON /IVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1                INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
     2                LNKSUF,SNKSUF,INSTYP
      COMMON /IVRFGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1                HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2                DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /IVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /IVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1                FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2                FLAG15
      COMMON /IVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1                MSA,PAGLOC,CURENT
      COMMON /IVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1                INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /IVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1                USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /IVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
     1                DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2                FINDFL,DLETFL,NSRTFL
      COMMON /IVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1                LUN
      COMMON /IVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /IVINS1/ REORG,FULL,RPLACE
      COMMON /IVVTR1/ NODSPC(1)
     1    /IVVTR2/ LSTSPC(1)
     2    /IVVTR3/ LNKSPC(1)
     3    /IVVTR4/ FLGSPC(1)
C
C THIS ROUTINE SEARCHES THE LIST TO FIND THE VALUE IN INCLUD.  IF IT
C IS FOUND, ITS POSITION WRT THE TOP OF THE LIST IS RETURNED
C
C DOES THE LIST EXIST ?
D     PAUSE 'IN LVINCL'
      IPOS = 1
      REQPAG(2) = -2
      J = 0
      CALL LVFDEX(J,J,J,J,J)
      IF(ITESTR .LT. 0) GO TO 31
      IF(IVAL .EQ. INCLUD) GO TO 25
      IF(LSTHED .LT. 0) GO TO 31
C MVL FOUND
      JVAL = IVAL
      KSKIP = NSKIP
      NSKIP = 0
      INDEX = 0
      INDXAD = 0
      KFUNC = 0
      KARG = 0
      SAVCON = 0
      KPOS = 0
 10   KPOS = KPOS+1
      IPOS = KPOS
      CALL LVFDEX(INDEX,INDXAD,KFUNC,KARG,SAVCON)
      IF(ITESTR .LT. 0) GO TO 30
      IF(IVAL .NE. INCLUD) GO TO 10
C
C EXIT FROM LOOP,  INCLUD =  1, SUCCESS
C          INCLUD = -1, FAILURE
C EXCEPT FOR IPOS, OUTPUT MUST APPEAR AS IF IT WAS FROM LVFIND
 30   IVAL = JVAL
      NSKIP = KSKIP
      LOC = LSTHED
 25   INCLUD = ITESTR
      ITESTR = 1
      RETURN
C
C SVL FAILURE RETURN
 31   INCLUD = -1
      RETURN
      END
```

```
C
C
C
         SUBROUTINE LVOVER
         IMPLICIT INTEGER(A-Z)
         LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR.DL1STR,
        1       DL2STR,DUMPFL,CURENT,FINDFL,DLETFL,NSRTFL,FD1TMP,
        2       DL2TMP,IN2TMP,FD2TMP,INSIDE,FULL,REORG,LSTCON.RPLACE,
        3       FINISH
         COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
        1              INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
        2              LNKSUF,SNKSUF,INSTYP
         COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
        1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
        2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
         COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
         COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK.FL3MSK,FL4MSK.FL5MSK,FLG67,
        1              FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
        2              FLAG15
         COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
        1              MSA,PAGLOC,CURENT
         COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
        1              INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
         COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
        1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
         COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN.IN1STR,IN2STR,FD1STR,FD2STR,
        1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
        2              FINDFL,DLETFL,NSRTFL
         COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
        1              LUN
         COMMON /LVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
         COMMON /LVFND/        COUNT,ABSPOS,LSTCON
         COMMON /LVINS1/ REORG,FULL,RPLACE
         COMMON /LVVTR1/ NODSPC(1)
        1    /LVVTR2/ LSTSPC(1)
        2    /LVVTR3/ LNKSPC(1)
        3    /LVVTR4/ FLGSPC(1)
C
D        PAUSE 'IN LVOVER'
C
C THIS ROUTINE HANDLES CONTINUANT OVERFLOW ON INDEXED INSERTION
C
C IF FUNCTION DOES NOT EXIST, RETURN AND PLACE ON NEXT CONTINUANT
          IF(ITESTR .LT. 0) RETURN
C
C PICK UP LAST VALUE ON LIST,
          IF(LSTHED .GT. 0) GO TO 50
C SVL
         NVAL = 2
         IVALS(2) = IVALS(1)
         ITYP1(2) = ITYP1(1)
         TMPVAL = LSTSPC(CTRL1+THIS)
         TMPTYP = FLGSPC(CTRL1+THIS) .AND. FLG67
         GO TO 100
C MVL
C SAVE THE VALUE TO BE INSERTED
50       CALL LVSTAC
```

```
      TMPVAL = NODSPC(CTRL1+LASTLC)
      TMPTYP = FIGSPC(CTRL1+LASTLC) .AND. FIG67
C
C DELETE LAST VALUE,
100   INDXON = 1
      IPOS = -1
      CALL LVDLET
C
C RESET CONTINUANT USAGE RATIO
      LNKSPC(CTRLPT+INSDEL) = LNKSPC(CTRLPT+INSDEL) -1
      IF(LSTHED .LT. 0) GO TO 150
C
C INSERT ORIGINAL VALUE,
      CALL LVPOP
      CALL LVNSRT
C
C INSERT LAST VALUE ON NEXT CONTINUANT
150   IVALS(1) = TMPVAL
      ITYP1(1) = TMPTYP
      INDXON = 0
      RETURN
      END
      C
      C
      C
            SUBROUTINE LVDUMP(DUMP)
            IMPLICIT INTEGER(A-Z)
            LOGICAL*1 CURENT,SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
           1            DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
           2            FINDFL,DLETFL,NSRTFL
            COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
           1            HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
           2            DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
            COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
            COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
           1            MSA,PAGLOC,CURENT
            COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
           1             INCORE,HDRSZF,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
            COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
           1            USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
            COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR.
           1            DL1STR.DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
           2            FINDFL,DLETFL,NSRTFL
            COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
           1            LUN
            COMMON /LVVSEQ/ ISEQSZ,ISOPOS,LASTSO,SEQSPC(1)
            COMMON /LVVTR1/ NODSPC(1)
           1    /LVVTR2/ LSTSPC(1)
           2    /LVVTR3/ LNKSPC(1)
           3    /LVVTR4/ FLGSPC(1)
            EQUIVALENCE(LUN.CHAN)
      C
      C THIS ROUTINE HAS TWO MODES OF OPERATION:
      C   A) BCD DUMP - OUTPUTS SYSTEM VARIABLES AND INTERNAL STRUCTURE OF GIRS
      C      PAGES AS A DEBUGGING AID
      C   B) BINARY DUMP
      C       COPIES ALL (MODIFIED) IN-CORE CONTINUANTS TO THE NEW DISK FILE.
      C       A CALL TO LVCLOS SAVES THE SYSTEM PARAMETERS.
      C
      C PAGES = -1, DUMP IN-CORE PAGE-BLOCKS
      C     =  0, DUMP ALL PAGES W/ ALL CONTS
      C     =  N. DUMP PAGE N
      C
      C IS THIS A BINARY OR BCD DUMP ?
      D     PAUSE 'IN LVDUMP()'
            IF(MODE .EQ. BINARY) GO TO 100
      C
      C*** BCD
      C LOGICAL UNIT NUMBER SHOULD BE DEFINED IN A CALL TO ASSIGN
      C WRITE HEADERS
            WRITE(LUN,10)
      10    FORMAT(/,' GIRS MEMORY DUMP (IN OCTAL)',//)
            ERRNUM = -1
            DUMPFL = .TRUE.
            CALL LVERR(DUMP)
            WRITE(LUN,21)
      21    FORMAT(///)
            WRITE(LUN,21)
      C WRITE IN-CORE DIRECTORY
      WRITE(LUN,35)
35    FORMAT('    IN-CORE DIRECTORY',//)
      WRITE(LUN,45)
45    FORMAT(1X,'     COUNTERS  KEY(PAGE)  CONTINUANT  "LOC-1"  LNKSPC
     1  FLGSPC',//)
      NBIAS = -HDRSZE
      NUMBLK = 1
      CALL LVWRIT(NBIAS,NUMBLK)
C
C TYPE OF DUMP ?
      IF(PAGES .GE. 0) GO TO 50
C WRITE OUT ONLY THOSE CONTINUANTS THAT ARE IN CORE
      NBIAS = DIRSZE
      NUMBLK = INCORE
      CALL LVWRIT(NBIAS,NUMBLK)
      DUMPFL = .FALSE.
      RETURN
C
C TEST TO WRITE OUT EITHER A SINGLE PAGE AND ALL OF ITS CONTINUANTS
C OR ALL PAGES
50    NPAGE = PAGES
      HIPAGE = HACTPG(1)
      IF(HREQPG .GT. HIPAGE) HIPAGE = HREQPG
      IF(PAGES.NE.0) GO TO 60
55    PAGES = PAGES+1
60    REQPAG(1) = PAGES
      REQPAG(2) = -1
65    REQPAG(2) = REQPAG(2)+1
C ATTEMPT TO LOCATE OR BRING IN PAGE, CONT
      CALL LVEXCH
C IS REQ(P,C) IN CORE ?
      IF(PAGLOC .GT. 0) GO TO 68
C ANY MORE CONTINUANTS TO THIS PAGE
      IF(MSARET .LE. 0) GO TO 70
C WRITE OUT PAGE
68    NBIAS = CTRLPT
      NUMBLK = 1
      CALL LVWRIT(NBIAS,NUMBLK)
      GO TO 65
C WRITE SINGLE PAGE ONLY ?
70    IF(NPAGE .NE. 0) GO TO 80
C IS THIS THE HIGHEST ACTIVE PAGE ?
      IF(PAGES .LT. HIPAGE) GO TO 55
80    DUMPFL = .FALSE.
      RETURN
C
C ***BINARY WRITE
C
100   CTRLPT = DIRSZE
      CTRL1  = CTRLPT + HDRSZE
        DO 150 I = 1, INCORE
C COPY TO DISK IF CONTINUANT HAS BEEN MODIFIED IN CORE
        IF((FLGSPC(CTRLPT + HDRFLG) .AND. MWRITE) .EQ. 0) GO TO 145
        FLGSPC(CTRLPT + HDRFLG) = 0
        LENGTH = PAGHDR
        BUFLOC = CTRLPT + 1
        MSA = NODSPC(CTRLPT + THSMSA)

        CALL LVPAGW
145     CTRLPT = CTRLPT + PAGHDR
        CTRL1  = CTRLPT + HDRSZE
150     CONTINUE
      CALL LVCLOS
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVWRIT(NBIAS,NUMBLK)
      IMPLICIT INTEGER(A-Z)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1               MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1               INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1               FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2               FLAG15
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1               USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1               LUN
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF.MASKPF
      COMMON /LVVTR1/ NODSPC(1)
     1    /LVVTR2/ LSTSPC(1)
     2    /LVVTR3/ LNKSPC(1)
     3    /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE PERFORMS A BCD WRITE OF NUMBLK PAGE-BLOCKS BEGINNING AT
C GIRS BUFFER LOCATION NBIAS
C
C FLAG CONTINUANT AS USED
      FLGSPC(CTRLPT+HDRFLG) = FLGSPC(CTRLPT+HDRFLG) .AND. .NOT. NOTUSD
D     PAUSE 'IN LVWRIT'
        DO 100 M = 1,NUMBLK
        ISTLOC = NBIAS+HDRSZE+1
        LAST = ISTLOC+PAGSZE-1
        IF(NBIAS .GT. 0) GO TO 10
        LAST = DIRSZE
        GO TO 30
C EXTRACT HEADER VALUES
10      MSA        = NODSPC(NBIAS + THSMSA)
        REGASP = NODSPC(NBIAS + REGAS)
        PAGE       = LSTSPC(NBIAS + PAGENO)
        CONT       = LSTSPC(NBIAS + CONTNO)
        INSDLT = LNKSPC(NBIAS + INSDEL)
        USEAGE = LNKSPC(NBIAS + USECT)
        FLAGS      = FLGSPC(NBIAS + HDRFLG)
        RDVAL      = FLGSPC(NBIAS + READVL)
        WRITE(LUN.1)
        WRITE(LUN,2) PAGE,CONT
C WRITE HEADER
        WRITE(LUN,3)
        WRITE(LUN,4) MSA,REGASP,INSDLT,USEAGE,FLAGS,RDVAL
        WRITE(LUN,5)
30      COUNT = 0
      DO 50 BUFCNT = ISTLOC, LAST
      COUNT = COUNT + 1
      M1 = NODSPC(BUFCNT)
      M2 = LSTSPC(BUFCNT)
      M3 = LNKSPC(BUFCNT)
      M4 = FLGSPC(BUFCNT)
      IF(NBIAS .GT. 0) GO TO 45
```

```
C IN-CORE DIRECTORY
          CONTIN = -1
          IF((FLGSPC(BUFCNT) .AND. FL1MSK) .EQ. 0) GO TO 40
          CONTIN = BUFCNT - M1 - 1
          IF(CONTIN .LT. 0) CONTIN = CONTIN + DIRSZE
          GO TO 47
40        M1 = 0
          M2 = 0
47        WRITE(LUN,7) BUFCNT,COUNT,M1,CONTIN,M2,M3,M4,COUNT
          GO TO 50
45        WRITE(LUN,6) BUFCNT,COUNT,M1,M2,M3,M4,COUNT
50        CONTINUE
           NBIAS = NBIAS+PAGHDR
100        CONTINUE
1         FORMAT(///,10X,'PAGE',3X,'CONTINUANT',/)
2         FORMAT(8X,I6,3X,I6,//)
3         FORMAT(1X,' MSA      REGASP    (INSERTIONS-DELETIONS)    USAGE
         1    FLAGS   READ COUNT')
4         FORMAT(1X,I5,2X,I5,10X,I5,14X,I5,3X,O3,3X,I5,///)
5         FORMAT(1X,'WRKSPC                    NOPSPC  LSTSPC  LNKSPC  FLGSPC
         1  OCTAL COUNTER',//)
6         FORMAT(1X,I6,2X,I6,2X,O6,2X,O6,2X,O6,2X,O6,2X,O6)
7         FORMAT(1X,I6,2X,I6,2X,O6,6X,I3,5X,O6,2X,O6,2X,O6,2X,O6)
          RETURN
          END
```

```
C
C
C
      SUBROUTINE LVCLOS
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVRAND/ PRIME,SEED,NROW,DNODE,DROW,OLDNOD,LISTSZ,
     1             GRNTBL(256)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1             MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1             INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1             USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVVSEQ/ ISEQSZ,ISQPOS,LASTSQ,SEQSPC(1)
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1             LUN
      COMMON /LVUSER/ USER(224)
D     PAUSE 'IN LVCLOS'
C
C SAVE SYSTEM VARIABLES ON FIRST BLOCK OF DISK
      RWBUF( 1)= REGASP
      RWBUF( 2)= NXTMSA
      RWBUF( 3)= PAGSZE
      RWBUF( 4)= PAGHDR
      RWBUF( 5)= BUFSZE
      RWBUF( 6)= DIRSZE
      RWBUF( 7)= DREGSP
      RWBUF( 8)= INCORE
      RWBUF( 9)= HDRSZE
      RWBUF(10)= HREQPG
      RWBUF(11)= HACTPG(1)
      RWBUF(12)= HACTPG(2)
      RWBUF(13)= READCT
      RWBUF(14)= BLKSZE
      RWBUF(15)= SUFSZE
      RWBUF(16)= DIRBLK
      RWBUF(17)= PRIME
      RWBUF(18)= SEED
      RWBUF(19)= LISTSZ
      RWBUF(20)= ISEQSZ
       DO 10 I = 1, 4
       RWBUF(20+I) = CURPAG(I)
       RWBUF(24+I) = LSTVPG(I)
10     CONTINUE
C
C USER WILL HAVE ACCESS TO WORDS 29 THRU 256 OF THE FIRST BLOCK TO STORE
C VARIABLES IF A PERMANENT FILE IS TO BE CREATED.
       DO 15 I = 29,256
       J = I - 28
15     RWBUF(I) = USER(J)
      LENGTH = 256
      MSA = 0
      ERRNUM = 3
      IERR = IWRITW(LENGTH,RWBUF(1),MSA,NWCHAN)
      DUMP = 0
      IF(IERR.LT.0) CALL LVERR(DUMP)
C
C SAVE GRN VARIABLES
C
      LENGTH = 256
      MSA = 1
      ERRNUM = 4
      IERR = IWRITW(LENGTH,GRNTBL(1),MSA,NWCHAN)
      DUMP = 0
      IF(IERR.LT.0) CALL LVERR(DUMP)
C
C SAVE INCORE DIRECTORY BEGINNING AT MSA = 2
C
      MSA = 2
      BUFLOC = 1
      LENGTH=DIRSZE
      ERRNUM = 5
      CALL LVPAGW
C
C SAVE CURRENT OUTCORE DIRECTORY
      CALL LVDRWR
C
C CLOSE CHANNEL
      CALL CLOSEC(NWCHAN)
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVPAGR(CHAN)
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSARET,
     1                HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2                DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DRFGSP,
     1                MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1                INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY.BCD,MODE,PAGES,
     1                LUN
      COMMON /LVMASK/ KWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF.MASKPF
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1                USECT,HDRFLG.READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVVTR1/ NODSPC(1)
     1     /LVVTR2/ LSTSPC(1)
     2     /LVVTR3/ LNKSPC(1)
     3     /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE READS DATA FROM DISK INTO RWBUF AND PLACES IT INTO WRKSPC
D     PAUSE 'IN LVPAGR()'
      NEWLEN = 4*LENGTH
      IERR =  IREADW(NEWLEN,RWBUF(1),MSA,CHAN)
      ERRNUM = 8
      DUMP = 0
      IF(IERR.LT.0) CALL LVERR(DUMP)
      ISTLOC = BUFLOC - 1
       DO 10 I = 1,LENGTH
       NODSPC(ISTLOC + I) = RWBUF(I)
       LSTSPC(ISTLOC + I) = RWBUF(LENGTH + I)
       LNKSPC(ISTLOC + I) = RWBUF(2*LENGTH + I)
10     FLGSPC(ISTLOC + I) = RWBUF(3*LENGTH + I)
      IF(ISTLOC .LE. 0) RETURN
C
C IF NOT DIRECTORY, FLAG CONTINUANT AS NOT USED
      FLGSPC(ISTLOC+HDRFLG) = FLGSPC(ISTLOC+HDRFLG) .OR. NOTUSD
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVPAGW
      IMPLICIT INTEGER(A-Z)
      LOGICAL*1 CURENT
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSAREF,
     1                HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2                DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
     1                MSA,PAGLOC,CURENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1                INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1                LUN
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
     1                USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSF,MASKPF
      COMMON /LVVTR1/ NODSPC(1)
     1      /LVVTR2/ LSTSPC(1)
     2      /LVVTR3/ LNKSPC(1)
     3      /LVVTR4/ FLGSPC(1)
C
C THIS ROUTINE TRANSFERS THE CONTENTS OF WRKSPC TO RWBUF TO BE WRITTEN
C OUT TO DISK
D     PAUSE 'IN LVPAGW'
      NEWLEN = 4*LENGTH
      ISTLOC = BUFLOC - 1
C
C IF NOT DIRECTORY, TURN FLAGS OFF
      IF(ISTLOC .LE. 0) GO TO 5
      FLGSPC(ISTLOC+HDRFLG) = FLGSPC(ISTLOC+HDRFLG) .AND. .NOT. FLGMSK
5      DO 10 I = 1,LENGTH
       RWBUF(I) = NODSPC(ISTLOC + I)
       RWBUF(LENGTH + I) = LSTSPC(ISTLOC + I)
       RWBUF(2*LENGTH + I) = LNKSPC(ISTLOC + I)
10     RWBUF(3*LENGTH + I) = FLGSPC(ISTLOC + I)
      IERR = IWRITW(NEWLEN,RWBUF(1),MSA,NWCHAN)
      ERRNUM = 9
      DUMP = 0
      IF(IERR.LT.0) CALL LVERR(DUMP)
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVDRRD(CHAN)
      IMPLICIT INTEGER(A-Z)
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSAREL,
     1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGBLK,BUFSZE,DIRSZE,
     1             INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHDR
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1             LUN
C
C THIS ROUTINE READS A SELECTED OUTCORE DIRECTORY BLOCK INTO OUTDIR()
C
D     PAUSE 'IN LVDRRD()'
      LENGTH = 256
      MSA = MSADIR + DIRBLK + DIRPAG
      ERRNUM = 6
      IERR = IREADW(LENGTH,OUTDIR(1),MSA,CHAN)
      DUMP = 0
      IF(IERR.LT.0) CALL LVERR(DUMP)
      RETURN
      END
C
C
C
      SUBROUTINE LVDRWR
      IMPLICIT INTEGER(A-Z)
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSAREL,
     1             HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2             DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE
     1             INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1             LUN
C
C THIS ROUTINE WRITES THE OUTCORE DIRECTORY BLOCK FROM OUTDIR() TO DISK
C
D     PAUSE 'IN LVDRWR'
      LENGTH = 256
      MSA = MSADIR + DIRBLK + DIRPAG - 1
      ERRNUM = 7
      IERR = IWRITW(LENGTH,OUTDIR(1),MSA,NWCHAN)
      DUMP = 0
      IF(IERR.LT.0) CALL LVERR(DUMP)
      RETURN
      END
```

```
C
C
C
      SUBROUTINE LVERR(DUMP)
      IMPLICIT INTEGER (A-Z)
      REAL*4 CORE
      LOGICAL*1 SNGLBK,SETUP,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
     1        DL2STR,DUMPFL,CTRENT,IN2TMP,FD2TMP,DL2TMP,FINDFL,
     2        DLETFL,SSRTFL,FULL,REORG,RPLACE
      COMMON /LVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
     1              INCLUD,INDXON,IVALS(10),ITYPE(10),SRCSIZ,
     2              LNKSIZ,SNKSIZ,INSTYP
      COMMON /LVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSAREL,
     1              HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
     2              DIRCNT,OUTLOC,OUTDIR(256),RWBUF(256)
      COMMON /LVMASK/ MWRITE,NOTUSD,NEWCON,FLGMSK,MASKSE,MASKPE
      COMMON /LVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
     1              FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
     2              FLAG15
      COMMON /LVRAND/ PRIME,SEED,NROW,DNODE,DROW,OLDNOD,LISTSZ,
     1              GRSTBL(256)
      COMMON /LVCRNT/ REGASP,CTRLPT,CTRL(3),LEASTV,NTFREE,FREE,DREGSP,
     1              MSA,PAGL(8),CTRENT
      COMMON /LVBUFR/ PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
     1              INCORE,HDRSZE,MSADIR,STFSZE,BLKSZE,DIRBLK,PAGPD4
      COMMON /LVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSPEL,
     1              USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
      COMMON /LVSWIT/ SETUP,SNGLBK,NXTRAS,IN1STR,IN2STR,FD1STR,FD2STR,
     1              DL1STR,DL2STR,IN2TMP,FD2TMP,DL2TMP,DUMPFL,
     2              FINDFL,DLETFL,SSRTFL
      COMMON /LVVSEQ/ ISEQSZ,ISQPOS,LASTSQ,SEQSP(2)
      COMMON /LVPRAM/ BUFLOC,LENGTH,IERR,ERRNUM,BINARY,BCD,MODE,PAGES,
     1              LUN
      COMMON /LVADDR/ IADD,THIS,LSTHED,LOC,LAST,LASTLC
      COMMON /LVSTAK/ CTRLEV,NUMVAR,STACK(400)
      COMMON /LVINST/ REORG,FULL,RPLACE
      COMMON /LVRUN/           RUNTYP,CORE
C
      DATA PART /1/
C
C   THIS ROUTINE IS USED TO PINPOINT DISK I/O ERRORS
C
 1    PAUSE ' IN LVERR DUMP'
      IF (ERRNUM .LT. 0) GO TO 4
      TYPE 1, ERRNUM,IERR
      FORMAT (24H **** DISK I/O ERROR NO. ,I3,5X,'ERRVAL = ',I3,/,
     1  'FULL DUMP PLACED ON ',5X,'ERROR ERR')
 2    FORMAT (10(1X,I6))
 3    FORMAT (10(1X,O6))
 4    FORMAT ( )
 5    FORMAT (10(2X,I))
 6    FORMAT (8(2X,I6))
 9    WRITE (LUN,10)
 10   FORMAT (' IADD THIS LSTHED,LOC LAST LASTLC')
      WRITE (LUN,2) IADD,THIS,LSTHED,LOC,LAST,LASTLC
      WRITE (LUN,4)
      WRITE (LUN,7)
```

```
11       FORMAT(' IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,INCLUD,
        1INDXON'/,
        1 'IVALS(1),ITYP1(1),SRCSUF,LNKSUF,SNKSUF,INSTYP')
         WRITE(LUN, 2)
        1 IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,INCLUD,INDXON,
        1 IVALS(1),ITYP1(1),
        2 SRCSUF,LNKSUF,SNKSUF,INSTYP
         WRITE(LUN, 3)
        1 IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,INCLUD,INDXON,
        1 IVALS(1),ITYP1(1),
        2 SRCSUF,LNKSUF,SNKSUF,INSTYP
         WRITE(LUN, 4)
         WRITE(LUN, 12)
12       FORMAT(' CURPAG(4)',30X,'REQPAC  )'./' LSTVPG(4)')
         WRITE(LUN,6) CURPAG,REQPAG,LST   G
         WRITE(LUN, 4)
         WRITE(LUN,122)
122      FORMAT(' MSARFT,HREQPG,NXTMSA,HACTPG(1),HACTPG(2),RFADCT,USECNT,
        2DIRPAG,DIRCNT,OUTLOC')
         WRITE(LUN, 2)
        1 MSARFT,HREQPG,NXTMSA,HACTPG,RFADCT,USECNT,DIRPAG,
        2 DIRCNT,OUTLOC
         WRITE(LUN, 4)
         WRITE(LUN, 13)
13       FORMAT(' PRIME,SEED,NROW,DNODE,DROW,OLDNOD,LISTSZ')
         WRITE(LUN, 2) PRIME,SEED,NROW,DNODE,DROW,OLDNOD,LISTSZ
         WRITE(LUN, 4)
         WRITE(LUN, 14)
14       FORMAT(' REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,MSA,
        1PAGLOC')
         WRITE(LUN, 2)
        1 REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,MSA,
        1 PAGLOC
         WRITE(LUN, 4)
         WRITE(LUN, 15)
15       FORMAT(' PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
        1INCORE,HDRSZE,MSADIR'/' SUFSZE,BLKSZE,DIRBLK,PAGHD4')
         WRITE(LUN, 2)
        1 PAGSZE,NWCHAN,OLCHAN,CMPAND,PAGHDR,BUFSZE,DIRSZE,
        1 INCORE,HDRSZE,MSADIR,SUFSZE,BLKSZE,DIRBLK,PAGHD4
         WRITE(LUN, 4)
         WRITE(LUN, 16)
16       FORMAT(' SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,DL1STR,
        1DL2STR,FINDFL'/' DLETFL,NSRTFL')
         WRITE(LUN, 5) SETUP,SNGLBK,NXTRAN,IN1STR,IN2STR,FD1STR,FD2STR,
        1            DL1STR,DL2STR,FINDFL,DLETFL,NSRTFL
         WRITE(LUN, 4)
         WRITE(LUN, 17)
17       FORMAT(' FULL,REORG,CURENT')
         WRITE(LUN, 5) FULL,REORG,CURENT
         WRITE(LUN, 4)
         WRITE(LUN, 18)
18       FORMAT(' BUFLOC,LENGTH,IERR,ERRNUM')
         WRITE(LUN, 2) BUFLOC,LENGTH,IERR,ERRNUM
         IF(DUMP .EQ. PART) GO TO 30
         WRITE(LUN, 4)
         WRITE(LUN, 123)
```

```
123     FORMAT(' OUTDIR(256)')
        WRITE(LUN,6) OUTDIR
        WRITE(LUN, 4)
        WRITE(LUN, 124)
124     FORMAT(' RWBUF(256)')
        WRITE(LUN,6) RWBUF
        WRITE(LUN, 4)
        WRITE(LUN, 131)
131     FORMAT(' GRNTBL(256)')
        WRITE(LUN,6) GRNTBL
        WRITE(LUN, 4)
        WRITE(LUN, 19)
19      FORMAT(' STACK')
        WRITE(LUN, 2) STACK
30      WRITE(LUN, 4)
        IF(DUMPFL .EQ. .TRUE.) RETURN
        CALL CLOSEC(NWCHAN)
        STOP
        END
C
C
C
        SUBROUTINE LVRTRN
        IMPLICIT INTEGER(A-Z)
        LOGICAL*1 CURENT
        COMMON /IVARGS/ IFUNC,IARG,IPOS,ITYP,IVAL,NVAL,NSKIP,ITESTR,
       1            INCLUD,INDXON,IVALS(10),ITYP1(10),SRCSUF,
       2            LNKSUF,SNKSUF,INSTYP
        COMMON /IVREGS/ CURPAG(4),REQPAG(4),LSTVPG(4),MSAREF,
       1            HREQPG,NXTMSA,HACTPG(2),READCT,USECNT,DIRPAG,
       2            DIRCNT,OUTLOC,OUTDIR(256),RWBUF(1)
        COMMON /IVFLAG/ FL0MSK,FL1MSK,FL2MSK,FL3MSK,FL4MSK,FL5MSK,FLG67,
       1            FLAG8,FLAG9,FLAG10,FLAG11,FLAG12,FLAG13,FLAG14,
       2            FLAG15
        COMMON /IVCRNT/ REGASP,CTRLPT,CTRL1,LEASTV,NTFREE,FREE,DREGSP,
       1            MSA,PAGLOC,CURENT
        COMMON /IVHDVL/ THSMSA,REGAS,PAGENO,CONTNO,INSDEL,
       1            USECT,HDRFLG,READVL,OLDNDH,DNODEH,NROWH,DROWH
        COMMON /IVVTR1/ NODSPC(1)
       1    /IVVTR2/ LSTSPC(1)
       2    /IVVTR3/ LNKSPC(1)
       3    /IVVTR4/ FLGSPC(1)
C
C THE PURPOSE OF THIS ROUTINE IS TO "UNDEFINE" A NODE.
D       PAUSE 'IN LVRTRN'
        FLGSPC(CTRL1 + IVAL) = FLGSPC(CTRL1 + IVAL) .AND. .NOT.FL3MSK
        RETURN
        END
```

```
C
C
C
      FUNCTION LVRTSH(WORD,BITS)
      IMPLICIT INTEGER(A-Z)
C
C THIS FUNCTION PERFORMS A RIGHT LOGICAL SHIFT
C
      IF(BITS .EQ. 0) GO TO 10
      IF(BITS .GE. 16) GO TO 20
      LVRTSH = WORD / 2 ** (BITS)
      RETURN
 10   LVRTSH = WORD
      RETURN
 20   LVRTSH = 0
      RETURN
      END
C
C
C
      FUNCTION LVLFSH(WORD,BITS)
      IMPLICIT INTEGER(A-Z)
C
C THIS FUNCTION PERFORMS A LEFT LOGICAL SHIFT
C
      IF(BITS .EQ. 0) GO TO 10
      IF(BITS .GE. 16) GO TO 20
      LVLFSH = WORD * 2 ** (BITS)
      RETURN
 10   LVLFSH = WORD
      RETURN
 20   LVLFSH = 0
      RETURN
      END
```

```
COMMON BLOCK /IVARGS/   LENGTH 000104

IFUNC    000000   INTEGER*2 VARIABLE
IARG     000002   INTEGER*2 VARIABLE
IPOS     000004   INTEGER*2 VARIABLE
ITYP     000006   INTEGER*2 VARIABLE
IVAL     000010   INTEGER*2 VARIABLE
NVAL     000012   INTEGER*2 VARIABLE
NSKIP    000014   INTEGER*2 VARIABLE
ITESTR   000016   INTEGER*2 VARIABLE
INCLUD   000020   INTEGER*2 VARIABLE
INDXON   000022   INTEGER*2 VARIABLE
IVALS    000024   INTEGER*2 ARRAY (10)
ITYP1    000050   INTEGER*2 ARRAY (10)
SRCSUF   000074   INTEGER*2 VARIABLE
LNKSUF   000076   INTEGER*2 VARIABLE
SNKSUF   000100   INTEGER*2 VARIABLE
INSTYP   000102   INTEGER*2 VARIABLE

COMMON BLOCK /IVREGS/   LENGTH 001056

CURPAG   000000   INTEGER*2 ARRAY (4)
REQPAG   000010   INTEGER*2 ARRAY (4)
LSTVPG   000020   INTEGER*2 ARRAY (4)

NAME     OFFSET   ATTRIBUTES

MSARET   000030   INTEGER*2 VARIABLE
HREQPG   000032   INTEGER*2 VARIABLE
NXTMSA   000034   INTEGER*2 VARIABLE
HACTPG   000036   INTEGER*2 ARRAY (2)
READCT   000042   INTEGER*2 VARIABLE
USECNT   000044   INTEGER*2 VARIABLE
DIRPAG   000046   INTEGER*2 VARIABLE
DIRCNT   000050   INTEGER*2 VARIABLE
OUTLOC   000052   INTEGER*2 VARIABLE
OUTDIR   000054   INTEGER*2 ARRAY (256)
RWBUF    001054   INTEGER*2 ARRAY (1)

COMMON BLOCK /IVMASK/   LENGTH 000014

MWRITE   000000   INTEGER*2 VARIABLE
NOTUSD   000002   INTEGER*2 VARIABLE
NEWCON   000004   INTEGER*2 VARIABLE
FLGMSK   000006   INTEGER*2 VARIABLE
MASKSF   000010   INTEGER*2 VARIABLE
MASKPF   000012   INTEGER*2 VARIABLE

COMMON BLOCK /IVFLAG/   LENGTH 000036

FL0MSK   000000   INTEGER*2 VARIABLE
FL1MSK   000002   INTEGER*2 VARIABLE
FL2MSK   000004   INTEGER*2 VARIABLE
FL3MSK   000006   INTEGER*2 VARIABLE
FL4MSK   000010   INTEGER*2 VARIABLE
FL5MSK   000012   INTEGER*2 VARIABLE
FLG67    000014   INTEGER*2 VARIABLE
FLAG8    000016   INTEGER*2 VARIABLE
FLAG9    000020   INTEGER*2 VARIABLE
FLAG10   000022   INTEGER*2 VARIABLE
FLAG11   000024   INTEGER*2 VARIABLE
FLAG12   000026   INTEGER*2 VARIABLE
FLAG13   000030   INTEGER*2 VARIABLE
FLAG14   000032   INTEGER*2 VARIABLE
FLAG15   000034   INTEGER*2 VARIABLE

COMMON BLOCK /IVRAND/   LENGTH 001016

PRIME    000000   INTEGER*2 VARIABLE
SEED     000002   INTEGER*2 VARIABLE
NROW     000004   INTEGER*2 VARIABLE
DNODE    000006   INTEGER*2 VARIABLE
DROW     000010   INTEGER*2 VARIABLE
OLDNOD   000012   INTEGER*2 VARIABLE
LISTSZ   000014   INTEGER*2 VARIABLE
GRNTBL   000016   INTEGER*2 ARRAY (256)
```

```
COMMON BLOCK /LVCRNT/   LENGTH 000023

REGASP  000000  INTEGER*2 VARIABLE
CTRLPT  000002  INTEGER*2 VARIABLE
CTRL1   000004  INTEGER*2 VARIABLE
LEASTV  000006  INTEGER*2 VARIABLE
NTFREE  000010  INTEGER*2 VARIABLE

NAME    OFFSET  ATTRIBUTES

FREE    000012  INTEGER*2 VARIABLE
DREGSP  000014  INTEGER*2 VARIABLE
MSA     000016  INTEGER*2 VARIABLE
PAGLOC  000020  INTEGER*2 VARIABLE
CURENT  000022  LOGICAL*1 VARIABLE

COMMON BLOCK /LVBUFR/   LENGTH 000034

PAGSZE  000000  INTEGER*2 VARIABLE
NWCHAN  000002  INTEGER*2 VARIABLE
OLCHAN  000004  INTEGER*2 VARIABLE
CMPAND  000006  INTEGER*2 VARIABLE
PAGHDR  000010  INTEGER*2 VARIABLE
BUFSZE  000012  INTEGER*2 VARIABLE
DIRSZE  000014  INTEGER*2 VARIABLE
INCORE  000016  INTEGER*2 VARIABLE
HDRSZE  000020  INTEGER*2 VARIABLE
MSADIR  000022  INTEGER*2 VARIABLE
SUFSZE  000024  INTEGER*2 VARIABLE
BLKSZE  000026  INTEGER*2 VARIABLE
DIRBLK  000030  INTEGER*2 VARIABLE
PAGHD4  000032  INTEGER*2 VARIABLE

COMMON BLOCK /LVHDVL/   LENGTH 000030

THSMSA  000000  INTEGER*2 VARIABLE
REGAS   000002  INTEGER*2 VARIABLE
PAGENO  000004  INTEGER*2 VARIABLE
CONTNO  000006  INTEGER*2 VARIABLE
INSDEL  000010  INTEGER*2 VARIABLE
USECT   000012  INTEGER*2 VARIABLE
HDRFLG  000014  INTEGER*2 VARIABLE
READVL  000016  INTEGER*2 VARIABLE
OLDNDH  000020  INTEGER*2 VARIABLE
DNODEH  000022  INTEGER*2 VARIABLE
NROWH   000024  INTEGER*2 VARIABLE
DROWH   000026  INTEGER*2 VARIABLE

COMMON BLOCK /LVSWIT/   LENGTH 000020

SETUP   000000  LOGICAL*1 VARIABLE
SNGLBK  000001  LOGICAL*1 VARIABLE
NXTRAN  000002  LOGICAL*1 VARIABLE
IN1STR  000003  LOGICAL*1 VARIABLE
IN2STR  000004  LOGICAL*1 VARIABLE
FD1STR  000005  LOGICAL*1 VARIABLE
FD2STR  000006  LOGICAL*1 VARIABLE
DL1STR  000007  LOGICAL*1 VARIABLE
DL2STR  000010  LOGICAL*1 VARIABLE
IN2TMP  000011  LOGICAL*1 VARIABLE
FD2TMP  000012  LOGICAL*1 VARIABLE
DL2TMP  000013  LOGICAL*1 VARIABLE
DUMPFL  000014  LOGICAL*1 VARIABLE
FINDFL  000015  LOGICAL*1 VARIABLE
DLETFL  000016  LOGICAL*1 VARIABLE
NSRTFL  000017  LOGICAL*1 VARIABLE

COMMON BLOCK /LVVSEQ/   LENGTH 000010

ISEQSZ  000000  INTEGER*2 VARIABLE
ISOPOS  000002  INTEGER*2 VARIABLE
LASTSO  000004  INTEGER*2 VARIABLE
SEQSPC  000006  INTEGER*2 ARRAY (1)
```

```
COMMON BLOCK /LVPRAM/   LENGTH 000022

BUFLOC  000000  INTEGER*2 VARIABLE
LENGTH  000002  INTEGER*2 VARIABLE
IERR    000004  INTEGER*2 VARIABLE
ERRNUM  000006  INTEGER*2 VARIABLE
BINARY  000010  INTEGER*2 VARIABLE
BCD     000012  INTEGER*2 VARIABLE
MODE    000014  INTEGER*2 VARIABLE
PAGES   000016  INTEGER*2 VARIABLE
LUN     000020  INTEGER*2 VARIABLE

COMMON BLOCK /LVSTAK/   LENGTH 000006

CURLEV  000000  INTEGER*2 VARIABLE
NUMVAR  000002  INTEGER*2 VARIABLE
STACK   000004  INTEGER*2 ARRAY (1)

COMMON BLOCK /LVUTIL/   LENGTH 000126

FILSPC  000000  INTEGER*2 ARRAY (39)
DEFEXT  000116  REAL*4    ARRAY (2)

COMMON BLOCK /LVINS1/   LENGTH 000003

REORG   000000  LOGICAL*1 VARIABLE
FULL    000001  LOGICAL*1 VARIABLE
RPLACE  000002  LOGICAL*1 VARIABLE

COMMON BLOCK /LVRUN/    LENGTH 000006

RUNTYP  000000  INTEGER*2 VARIABLE
CORE    000002  REAL*4    VARIABLE

COMMON BLOCK /LVVTR1/   LENGTH 000002

NODSPC  000000  INTEGER*2 ARRAY (1)

COMMON BLOCK /LVVTR2/   LENGTH 000002

LSTSPC  000000  INTEGER*2 ARRAY (1)

COMMON BLOCK /LVVTR3/   LENGTH 000002

LNKSPC  000000  INTEGER*2 ARRAY (1)

COMMON BLOCK /LVVTR4/   LENGTH 000002

FIGSPC  000000  INTEGER*2 ARRAY (1)


COMMON BLOCK /LVDEL1/   LENGTH 000003

NUMRET  000000  INTEGER*2 VARIABLE
BAKCON  000002  LOGICAL*1 VARIABLE

COMMON BLOCK /LVADDR/   LENGTH 000014

IADD    000000  INTEGER*2 VARIABLE
THIS    000002  INTEGER*2 VARIABLE
LSTHED  000004  INTEGER*2 VARIABLE
LOC     000006  INTEGER*2 VARIABLE
LAST    000010  INTEGER*2 VARIABLE
LASTLC  000012  INTEGER*2 VARIABLE

COMMON BLOCK /LVFND/    LENGTH 000005

COUNT   000000  INTEGER*2 VARIABLE
ABSPOS  000002  INTEGER*2 VARIABLE
LSTCON  000004  LOGICAL*1 VARIABLE
```

REFERENCES

1. Zaritsky, I., "GIRS (Graph Information Retrieval System) Users Manual," DTNSRDC Report 79/036 (Apr 1979).

2. Berkowitz, S., "Design Trade-offs for a Software Associative Memory," DTNSRDC Report 3531 (May 1973).

3. Zaritsky, I., "Feasibility Study for Incorporating A Data Structure Definition and Manipulation Facility Within The COMRADE Data Management System," DTNSRDC Report 78/045 (May 1978).

4. Carlberg, J., "A Paged Hardware Associative Memory - Preliminary Report," DTNSRDC Report 77-0083 (Aug 1977).

5. Berkowitz, S., "Graph Information Retrieval Language; Programming Manual for FORTRAN Complement; Revision One," DTNSRDC Report 76-0085 (Feb 1976).

6. "Data Handler, Version 1.0, Reference Manual, Revision B," Control Data Corporation Publication No. 17322100 (Jan 1976).

INITIAL DISTRIBUTION

Copies	
1	CHONR
1	NRL
1	NSWC
1	NUSC
1	NOSC
1	NAVSUP/0431C, G. Bernstein
2	NAVSEA 1 SEA 312 1 SEA 612
1	Rome Air Development Center
12	DTIC

CENTER DISTRIBUTION

Copies	Code	Name
1	18	G. Gleissner
1	1802.2	F. Frenkiel
1	1803	S. Rainey
1	1804	L. Avrunin
1	1805	E. Cuthill
1	1806	R. Santamaria
2	1809.3	D. Harris
1	182	A. Camara
1	1821	D. Jefferson
1	1822	T. Rhodes
1	1824	S. Berkowitz
1	1824	J. Carlberg
1	1824	J. Garner
1	1824	P. Marques
1	1824	C. Slominski
20	1824	I. Zaritsky
1	1826	L. Culpepper
1	184	J. Schot
1	184.1	H. Feingold
1	1843	H. Haussling
1	1844	S. Dhir
1	1844	J. McKee
1	185	T. Corin
1	1850	A. Cinque
1	1851	J. Brainin
1	1854	H. Sheridan
1	1855	R. Brengs
1	187	M. Zubkoff
1	189	G. Gray
10	5211.1	Reports Distribution
1	522.1	Unclass Lib (C)
1	522.2	Unclass Lib (A)